André de Pádua Pereira

Caracterização das necessidades computacionais para perfis Argo e suas aplicações

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: João Carlos Gomes Moura Pires, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-orientadores: José Júlio Alferes, Professor Catedrático, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Carlos Viegas Damásio, Professor Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente: Nuno Preguiça
Arguente: Karine Ferreira
Vogal: João Pires

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Junho, 2019

Caracterização das necessidades computacionais para perfis Argo e suas aplicações

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Agradecimentos

Gostaria de agradecer,

ao meu orientador, João Moura Pires, por todo o apoio, recomendações e ajuda indispensável para a realização deste trabalho. Foi um prazer trabalhar sob a sua orientação.

aos meus co-orientadores, Carlos Damásio e José Alferes, que me acompanharam durante toda a dissertação, e ajudaram a tornar este trabalho o melhor possível.

à Altran Portugal, que tornou este projeto possível e permitiu a minha integração na sua equipa de investigação e desenvolvimento.

a todos os meus amigos, que fiz ao longo do meu percurso na FCT-UNL.

Resumo

Os oceanos compõem mais de 70 por cento da superfície do planeta, embora seja a fração menos estudada do mesmo, dada a sua extensão e as suas condições pouco favoráveis à sua observação.

O projeto Argo foi criado com o objetivo de registar sistematicamente a temperatura e salinidade da camada superior do oceano, até aos 2000 metros de profundidade. As observações são efetuadas por boias autónomas espalhadas pelo oceano, que a cada 10 dias descem até aos 2000m de profundidade e ascendem até à superfície efetuando medições a diferentes níveis de pressão, resultando num perfil vertical de temperatura e salinidade da coluna vertical da parte superior do oceano. Atualmente existem cerca de 4000 boias ativas que produzem mais de 100 000 perfis por ano.

Desde o inicio do Projeto Argo em 1999, surgiram inúmeros estudos sobre diferentes propriedades do oceano que utilizam como base os perfis medidos pelas suas boias. Alguns destes estudos são caracterizados por utilizar dados de diversas fontes complementadas com os perfis Argo, como por exemplo dados de satélite sobre o nível do mar.

O aumento do número de perfis a serem medidos a cada ano, as extensões planeadas ao projeto Argo que irão aumentar o leque de variáveis do oceano a serem medidas e a cobertura vertical dos perfis, e ainda a utilização conjunta destes perfis com dados de outras fontes por parte de diferentes estudos, incentivam o estudo de tecnologias *BigData* para armazenar e explorar estes dados de forma eficiente, para facilitar a sua análise cientifica.

O objetivo desta dissertação é, em primeiro lugar, caracterizar diferentes aplicações e requisitos computacionais associados a dados provenientes do projeto Argo. Além disso é escolhido um caso de estudo, de forma a explorar e avaliar o uso de diferentes tecnologias *Big Data* com foco em dados espaciais e espaciotemporais, no armazenamento e processamento dos diferentes dados associados.

Palavras-chave: Argo, Time Series, SpatioTemporal Big Data, Big Data, Argo profile applications

Abstract

The oceans make up for more than 70 percent of the planet's surface. They are also the least studied fraction of the planet, given its extent and its unfavorable conditions for measurements. Due to the lack of observational data with relevant spatial and temporal resolutions, the study of the ocean has always been a complex issue.

The Argo project was created with the goal of systematically recording the temperature and salinity of the upper layer of the ocean, up to 2000 meters deep. The observations are made by autonomous floats scattered across the ocean, which descend to 2000m depth and ascend to the surface every 10 days while taking measurements at different pressure levels, resulting in a vertical profile of temperature and salinity of the vertical column of the upper part of the ocean. Currently there are about 4000 active floats that produce more than 100,000 profiles per year.

Since the beginning of the Argo Project in 1999, there have been numerous studies on different properties of the ocean that use as a basis the profiles measured by its floats. Some of these studies are characterized by using data from several sources supplementing Argo profiles, such as satellite data on sea level.

The increasing number of profiles to be measured each year, the planned extensions to the Argo project that will increase the range of ocean variables to be measured and the vertical coverage of the profiles, as well as the joint use of these profiles with data from other sources by different studies, encourage the study of BigData technologies to efficiently store and exploit this data in order to facilitate its scientific analysis.

The purpose of this dissertation is to characterize different applications and computational requirements associated with data coming from the Argo project, and by choosing one use case, in order to explore and evaluate the use of different BigData technologies focusing on spatial and spatiotemporal data in the storage and processing of the different associated data.

Keywords: Argo, Time Series, SpatioTemporal Big Data, Big Data, Argo profile applications

m

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Siglas

AMOC	Atlantic Meridional Overturning circulation.
ENSO	El Nino Southern Oscilation.
HDFS	Hadoop Distributed File System.
MBR	Minimum Bounding Rectangle.
NetCDF	Network Common Data Form.
RDD	Resilient Distributed Dataset.
SRDD	Spatial Resilient Distributed Dataset.
YARN	Yet Another Resource Negotiator.

Capítulo 1

Introdução

1.1 Motivação e Objetivos

Presentemente existe uma crescente preocupação com os efeitos climáticos do fenómeno do aquecimento global no planeta Terra. Os oceanos compõem 70% da superfície terrestre e têm uma influência significativa no controlo do seu clima. O estudo das dinâmicas e fenómenos associados a esta parte do planeta são cruciais para compreender o seu impacto no clima no futuro[1]. A realização destes estudos sempre foi uma tarefa complicada, dada a vasta extensão dos oceanos, as condições pouco propicias para medições experimentais e também a falta de dados observacionais com resoluções espaciais e temporais relevantes.

O projeto Argo surge em 1999, através de uma cooperação internacional entre diversos países, para responder a essa necessidade, com a implementação de uma matriz de boias autónomas e automáticas no âmbito da medição da temperatura e salinidade da camada superior de 2000 metros do oceano. A matriz apresenta uma resolução espacial de $3° \times 3°$ (a distância entre duas boias é aproximadamente 300km) e cada boia efetua medições aproximadamente a cada 10 dias. Atualmente, o projeto Argo conta com 3878 boias ativas que fornecem mais de 100 000 perfis verticais de temperatura e salinidade do oceano por ano. A figura 1.1 apresenta a localização das boias ativas em maio de 2019 organizadas por país contribuidor.

[1] http://www.whoi.edu/know-your-ocean/

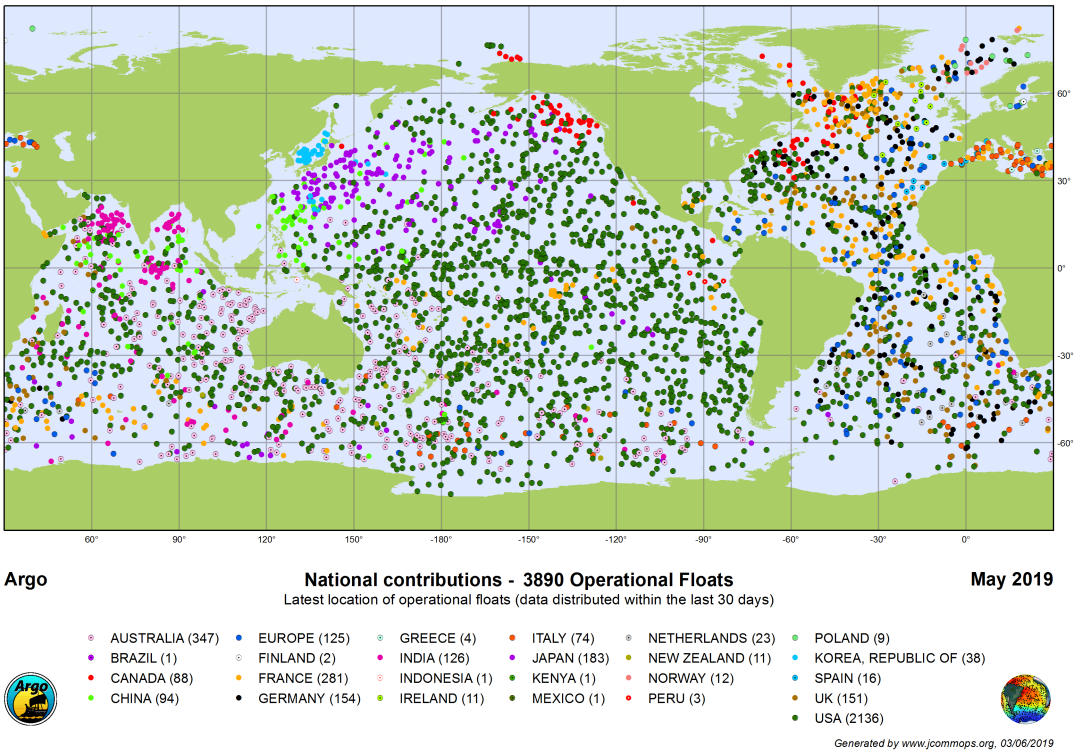


Figura 1.1: Localização das boias do projeto Argo em Maio de 2019 [2].

A disponibilidade de uma fonte de dados global e sistemática contribuiu para diversos estudos, e já foram publicados mais de 3500 artigos que utilizam essa fonte. A Figura 1.2 [1] apresenta o número de artigos publicados por ano que utilizam dados provenientes do projeto Argo.

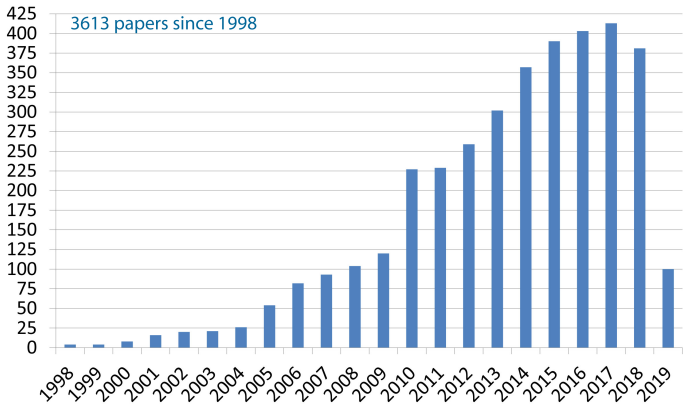


Figura 1.2: Número de publicações cientificas por ano, que utilizam dados Argo [1].

Estes artigos exploram diferentes propriedades e fenómenos presentes no oceano, não apenas com dados Argo, mas também com dados de outras fontes. A resolução global

[2] argo.jcommops.org, visitado em 05/2019

deste conjunto de dados permite que sejam feitos estudos a diferentes escalas espaciais, desde estudos em regiões específicas, a estudos sobre a globalidade dos oceanos.

Embora exista um conjunto extenso de literatura que explora diferentes aspetos do oceano, não existe ainda um estudo sistemático das necessidades computacionais das diferentes aplicações sobre estes dados. Um dos objetivos desta dissertação é o levantamento e caracterização desses requisitos computacionais. Para tal, foi necessário compreender em concreto quais são as aplicações relevantes sobre os dados e de que forma é que estes são utilizados.

Após o levantamento de requisitos, foi analisado o estado da arte em tecnologias Big Data para o processamento de dados espaciais / espaciotemporais. Como resultado do estudo, algumas tecnologias foram avaliadas na sua capacidade de processamento do conjunto de dados estudado.

Por fim, foi implementado um caso de uso relevante aos dados Argo utilizando algumas das tecnologias estudadas, que consiste num produto que organiza a informação de perfis numa grelha espaciotemporalmente uniforme.

Esta dissertação foi desenvolvida em contexto empresarial, inserida num projeto de Investigação & Desenvolvimento da Altran Portugal.

1.2 Metodologia de Trabalho

Neste documento está apresentado o trabalho efetuado durante o período de dissertação. O trabalho efetuado durante este período caracterizou-se pela análise de literatura relevante ao tema a ser abordado, participação em reuniões de grupo semanais com o foco de partilha de conhecimento e ainda por reuniões individuais com o orientador e co-orientadores da dissertação, para a discussão de resultados intermédios e *feedback* de trabalho realizado.

Para o cumprimento de todos os objetivos esperados, foram identificadas e seguidas as seguintes fases metodológicas que guiaram todo o trabalho realizado:

Identificação do problema e objetivos Identificar o problema e os objetivos associados ao tema a ser explorado.

Revisão de Literatura Revisão da literatura relacionada com o projeto Argo e tecnologias Big Data de forma a obter o maior conhecimento possível sobre os dados a serem utilizados e as ferramentas a avaliar.

Desenho experimental Definição de uma arquitetura experimental que responda aos objetivos de avaliação das diferentes tecnologias Big Data, sob a forma de resultados experimentais.

Discussão de resultados Discussão dos resultados experimentais obtidos.

1.2.1 Análise de Literatura

A literatura analisada durante o período de dissertação focou-se em dois temas distintos: Projeto Argo e Tecnologias Big Data.

1.2.1.1 Projeto Argo

A primeira análise sobre este tema procurou obter mais conhecimento sobre a história do projeto Argo, o seu funcionamento e as diferentes propriedades dos dados relacionados com este projeto. Para a análise destes pontos, a pesquisa foi feita através de informação disponível no *website* oficial do projeto [3].

O foco seguinte passou pela análise da bibliografia associada ao projeto. O objetivo da análise passou por obter informação de como os perfis Argo eram utilizados em trabalhos relacionados com o estudo do oceano, para poderem ser levantados requisitos computacionais e aplicações associadas.

Como a bibliografia relacionada com o projeto Argo é muito extensa, a abordagem inicial consistiu na leitura de artigos mais recentes com diferentes tipos de análises, de forma a obter uma visão mais expandida possível das diferentes aplicações dos dados.

A grande extensão da bibliografia dificultou a sua caracterização global, pelo que também foi abordada uma solução mais sistemática para a sua análise, que se encontra descrita no capítulo 2.

1.2.1.2 Big data

O estudo da literatura sobre este tema, que se descreve adiante na secção 3.3, surgiu como resposta à necessidade de processamento e análise de alguns casos de uso sobre os perfis Argo.

Após a obtenção de uma visão geral da área, o interesse foi o de procurar trabalhos/estudos/tecnologias com o foco em dados espaciotemporais, devido às características dos dados a serem explorados durante esta dissertação. Como complemento a este trabalho, foram também analisadas algumas técnicas de particionamento espacial utilizadas por alguns dos *frameworks* estudados.

1.3 Contribuições

As contribuições desta dissertação são as seguintes.

- Identificação das diferentes aplicações relacionadas com os dados provenientes do projeto Argo, através de uma análise sistemática da sua bibliografia.
- Análise dos requisitos computacionais associados às diferentes aplicações, com base na caracterização da bibliografia obtida.

[3] http://www.argo.ucsd.edu, visitado em 02/2018

- Identificação de diferentes tecnologias Big Data apropriadas para o armazenamento e processamento de dados do projeto Argo.
- Implementação de um caso de uso pertinente ao Projeto Argo, utilizando as tecnologias estudadas.

1.4 Estrutura do Documento

Este documento, contém seis capítulos e está estruturado da seguinte forma:

Projeto Argo Neste capítulo são descritos aspetos fundamentais do projeto Argo e é conduzido o levantamento de aplicações associadas aos seus dados, bem como a explicitação da metodologia seguida para a análise sistemática da sua bibliografia associada.

Trabalho Relacionado O terceiro capítulo contém o trabalho de pesquisa que complementou a análise do projeto Argo. É apresentada informação sobre tecnologias Big Data para processamento de dados espaciais e espaciotemporais. Contém também informação sobre algoritmos de divisão espacial, importantes para entender o funcionamento das tecnologias abordadas.

Análise Experimental Este capítulo apresenta o trabalho experimental realizado sobre as diferentes tecnologias *Big data* estudadas.

Implementação de Caso de Uso O sexto capítulo do documento, contém detalhes da implementação de um dos casos de uso apresentados anteriormente no documento. O caso de uso escolhido refere-se à criação do produto *ArgoMixedLayers*, que agrega diferentes propriedades das medições de boias Argo e as providencia sob a forma de uma grelha, facilitando a sua análise científica.

Conclusões e Trabalho Futuro Este capítulo contém um resumo e análise do trabalho efetuado e também das conclusões obtidas nesta dissertação.

No final do documento, estão ainda disponíveis em anexo dois capítulos:

- Um capítulo contendo toda a configuração da infraestrutura física utilizada no trabalho experimental realizado.
- Um capítulo sobre séries temporais, mais especificamente sobre diferentes representações. Este trabalho de pesquisa foi realizado devido à componente temporal de perfis Argo, mas foi constatado que devido à baixa complexidade temporal dos mesmos, não foi necessária a sua aplicação.

Capítulo 2

Projeto Argo

Este capítulo é dedicado ao projeto Argo e às suas particularidades. É apresentada a sua história, a constituição e ciclo de funcionamento das suas boias, a descrição das diferentes formas de acesso aos dados e uma breve apresentação das suas extensões em desenvolvimento.

Para além da descrição do projeto são apresentadas as diferentes aplicações dos dados Argo a descrição do processo da análise sistemática da sua bibliografia associada.

2.1 Projeto Argo

O projeto Argo foi iniciado em 1999 com o objetivo de obter uma visão global e sistemática da temperatura e salinidade do oceano. O conjunto de medições destas variáveis existentes até à altura provinham de sensores colocados em barcos. Este método de amostragem apresentava algumas desvantagens porque dependia de linhas marítimas e os dados obtidos eram tendenciosos, já que a maioria das amostras estavam concentradas no Hemisfério Norte e eram obtidas apenas durante as estações mais quentes do ano [2]. Para colmatar esse problema, o projeto Argo iniciou a criação de uma matriz global de boias com resolução espacial de $3° \times 3°$ (cada duas boias encontram-se aproximadamente a 300km entre si) de forma a cobrir globalmente o oceano. Para a matriz ter a resolução espacial pretendida, eram necessárias 3000 boias, número que foi atingido em 2007. A criação e manutenção deste projeto conta com grande cooperação internacional, e já mais de 30 países contribuíram com boias [1] .

Atualmente o projeto conta com 3890 boias ativas[2], número superior ao inicialmente pensado, devido à expansão da cobertura da matriz para áreas do oceano marginais,

[1] http://www.argo.ucsd.edu/Organisation.html, visitado a 02/2018
[2] http://www.jcommops.org/argo, visitado em 05/2019

onde a profundidade é menor que 2000m e em áreas onde existe formação de gelo sazonalmente. A figura 2.1, representa a cobertura das boias ativas em maio de 2019, ao apresentar o número de perfis medidos em cada divisão de 3 graus de latitude e longitude.

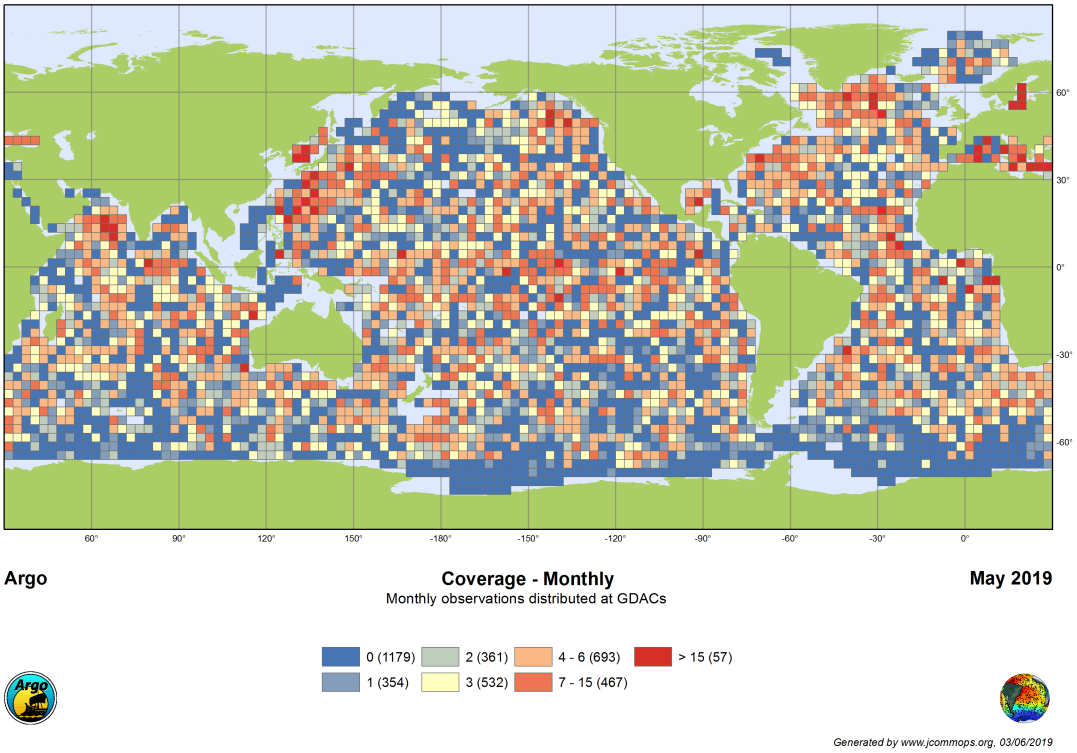


Figura 2.1: Localização das boias do projeto Argo em Maio de 2019 [2].

Outro foco principal deste projeto é a disponibilização em tempo real dos perfis obtidos pelas boias. Há uma estrutura terrestre de forma a suportar esse desígnio. Existem *Data Assembly Centers* (DAC) em diferentes pontos do globo, cujo propósito é receber os dados que provêm das boias, aplicar testes de controlo de qualidade automáticos [3] e enviar os dados para os *Global Data Assembly Center* (GDAC), onde os dados são disponibilizados de forma gratuita para qualquer utilizador. Existem dois centros de dados globais localizados em Brest (França) e Monterey (Estados Unidos). Os dados também são disponibilizados para centros operacionais através de um sistema de comunicação próprio, designado por *Global Telecommunication System (GTS)*. O fluxo dos dados desde a boia até ao utilizador final está ilustrado na Figura 2.2.

2.1.1 Boia Argo

Uma boia Argo funciona de forma autónoma e tem uma esperança média de vida de 4-5 anos. A figura 2.3 apresenta a esquemática de uma boia Argo. Cada boia contém um sensor CTD (*Conductivity, Temperature, Depth*) utilizado para medir a temperatura, salinidade e pressão. Os sensores utilizados nestas boias possuem uma precisão nas suas

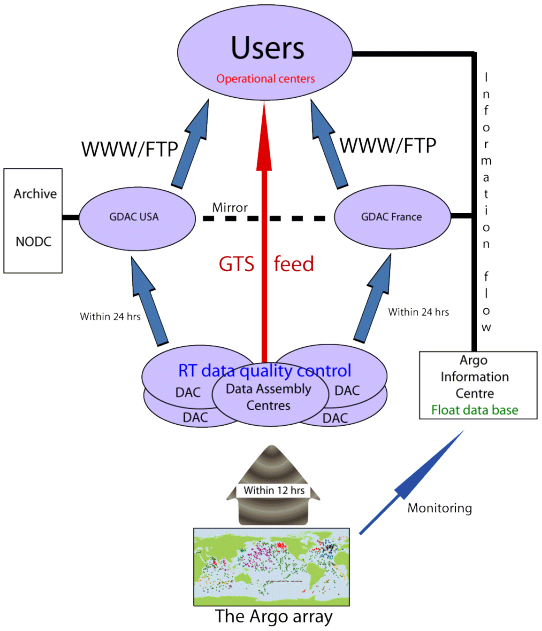


Figura 2.2: Representação do fluxo dos dados Argo disponibilizados em tempo real. Imagem retirada do website oficial do projeto Argo.

medições de 0.005 °C, 0.01 unidades de salinidade (PSU) e 2.5 dbars de pressão. Para além do sensor CTD, uma boia também possui um sistema de controlo de flutuabilidade que permite a sua movimentação vertical no oceano. Este sistema controla a densidade da boia através da expansão e contração de uma bolsa, o que permite a boia flutuar a diferentes níveis de profundidade. A expansão e contração da bolsa é feita através da inserção /remoção de um líquido que se encontra num reservatório dentro da boia. Por fim, cada boia possui uma antena que é utilizada para a determinação da sua localização à superfície do oceano e para a transmissão de dados via satélite. Atualmente são utilizados o sistema Argos [3] e Iridium [4] para a transmissão de dados da boia para os DAC.

2.1.2 Ciclo de medição de uma boia Argo

O processo efetuado por uma boia para a obtenção de um perfil vertical de uma área do oceano é designado por um ciclo de medição Argo.

O ciclo de medição de uma boia é iniciado quando esta se desloca da superfície até uma profundidade de 1000m, onde fica à deriva durante aproximadamente 9 dias. Após esse período, a boia desloca-se até uma profundidade de 2000m e inicia a sua ascensão até à superfície. Durante um período de 6 horas, a boia ascende até à superfície e regista a diferentes níveis de profundidade, os valores de temperatura, salinidade e pressão. Ao chegar à superfície, é determinada a sua localização através de satélites e as medições

[3] http://www.argos-system.org/ visitado em 02/2018
[4] https://www.iridium.com/
[5] http://www.argo.ucsd.edu/float_design.html, visitado a 05/2019

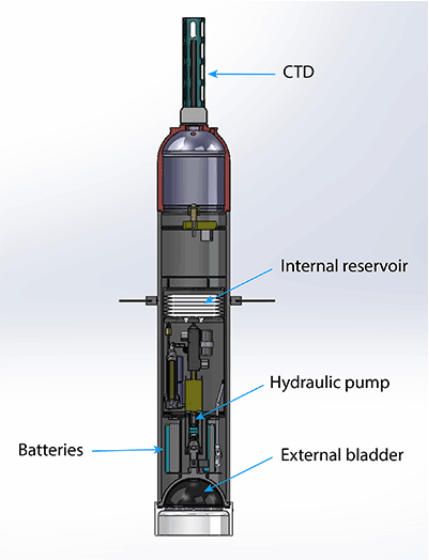


Figura 2.3: Esquemática de uma boia Argo [5].

obtidas são transmitidas para centros de dados terrestres. Este ciclo tem a duração média de 10 dias (Figura 2.4).

O número de medições feitas durante a ascensão da boia e o tempo que esta necessita de estar à superfície depende do sistema de satélites utilizado para a transmissão dos dados. Boias que utilizam o sistema Argos necessitam de estar entre 6 a 12h à superfície e estão limitadas a 200 medições durante a sua ascensão. Boias que utilizam o sistema Iridium para transmissão de dados e GPS (*Global Positioning System*) para a determinação da sua localização, necessitam apenas de alguns minutos à superfície para o envio dos dados, e poderão efetuar até 1000 medições durante a sua ascensão. Em maio de 2019, 71% das boias ativas utilizava a rede Iridium [2].

2.1.3 Acesso aos dados

Os dados contidos nos centros de dados globais são distribuídos de duas formas distintas:

Tempo Real: Dados disponíveis 24h após serem enviados por uma boia. Estes dados são apenas submetidos a controlos de qualidade automáticos [3] antes de serem disponibilizados, pelo que poderão conter algumas imprecisões.

Modo diferido: Dados disponíveis aproximadamente 1 ano após o seu envio. Estes dados contém correções devido a possíveis desajustes nos sensores das boias. Para a sua correção, os dados são avaliados por oceanógrafos experientes.

Ambos os centros de dados disponibilizam acesso direto a todos os perfis disponíveis, organizados por DAC, ou por oceano e data (mês e ano) em que o perfil foi medido. Os dados são disponibilizados em formato Network Common Data Form (NetCDF). Cada

[6] http://www.argo.ucsd.edu/How_Argo_floats.html, visitado a 05/2019

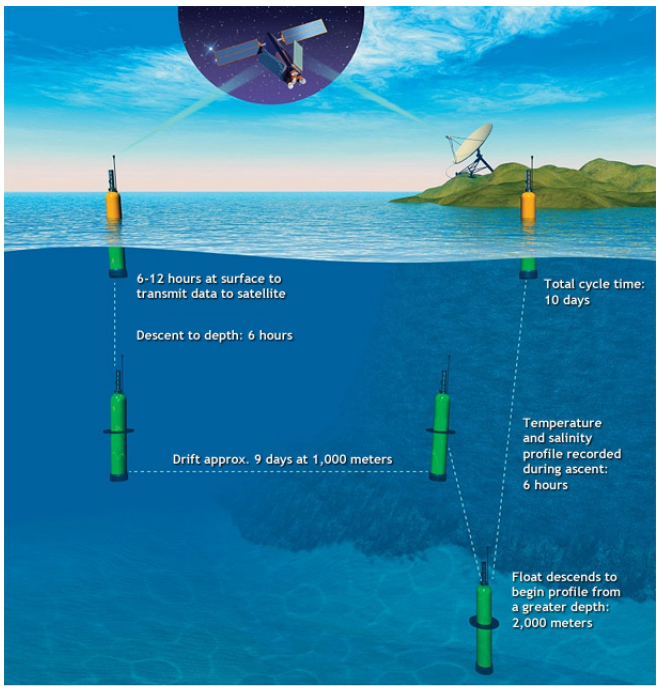


Figura 2.4: Ciclo de medição de uma boia Argo [6].

ficheiro contém informação sobre a boia, medições das diferentes variáveis (pressão, temperatura e salinidade) para os diferentes perfis e ainda contém informação sobre a trajetória percorrida pela boia. Uma descrição mais detalhada do formato dos ficheiros referidos pode ser encontrada em [4].

Para além do acesso direto aos ficheiros alocados nos centros, ambos disponibilizam ferramentas visuais de forma a ajudar na seleção de perfis. Estas ferramentas permitem o filtro de perfis por área, data ou seleção de boias [7] [8].

2.1.4 Exemplo de um ciclo

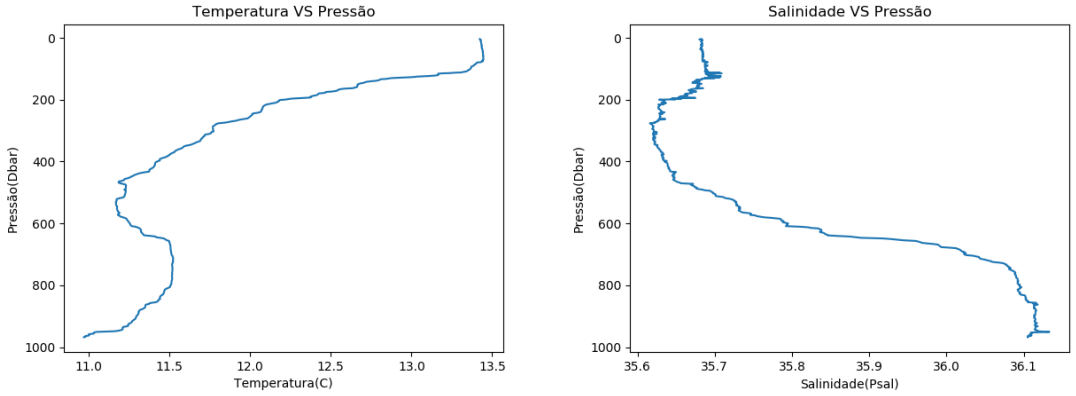
Para demonstrar um exemplo do ciclo de vida de uma boia Argo, são apresentadas de seguida, a trajetória efetuada pela boia WMO-3901909, que se trata da primeira boia contribuída por Portugal, no contexto do programa Euro-Argo[9], e alguma informação sobre perfis de temperatura e salinidade medidos por esta. Os dados da boia foram obtidos através da ferramenta de seleção de boias, que pode ser consultada em http://www.argodatamgt.org/Access-to-data/Description-of-all-floats2

A boia já se encontra ativa desde dezembro de 2016 e efetuou 41 medições de perfis. À data da última consulta, o último perfil medido por esta boia foi em 25/01/2018 e contém 968 medições de temperatura e salinidade ao longo dos 1000 metros superiores

[7] http://www.argodatamgt.org/Access-to-data/Argo-data-selection, visitado em 12/2017
[8] http://www.usgodae.org/cgi-bin/argo_select.pl, visitado a 12/2017
[9] https://www.ipma.pt/pt/media/noticias/news.detail.jsp?f=/pt/media/noticias/arquivo/2017/Euro-Argo.html, visitado a 12/2017

de profundidade do oceano. Este perfil contém apenas medições até aos 1000m devido à sua proximidade com o litoral português (como será possível observar na figura 2.6)

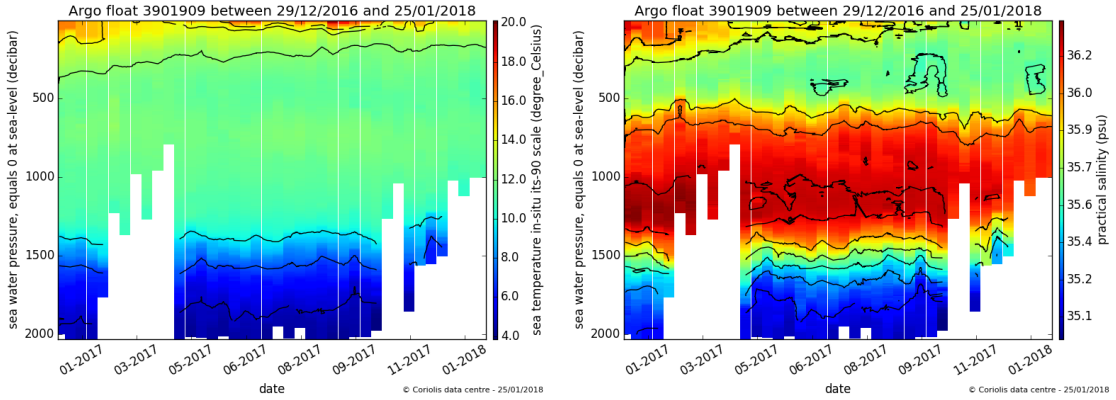
A figura 2.5 apresenta a temperatura e salinidade nas diferentes pressões do perfil:



(a) Tempratura (b) Salinidade

Figura 2.5: Gráficos de temperatura e salinidade vs pressão das medições do perfil nº 40 da boia WMO-3901909.

Cada perfil também contém as coordenadas (latitude e longitude) do local de onde foi retirado. A figura 2.6 apresenta as 40 localizações dos diferentes perfis medidos por esta boia. Por fim, a ferramenta de onde foram retirados os dados das boias também contém diferentes gráficos sobre os diferentes perfis. Na figura 2.7 são apresentados graficamente os 40 perfis medidos pela boia, em relação à temperatura e salinidade, onde é possível constatar as diferentes profundidades a que estes foram medidos.



(a) Temperatura (b) Salinidade

Figura 2.7: Gráficos que apresentam a temperatura(a) e salinidade(b) dos 40 perfis medidos pela boia WMO-3901909 [10].

[10]Retirados de http://www.argodatamgt.org/Access-to-data/Description-of-all-floats2, visitado a 02/2018

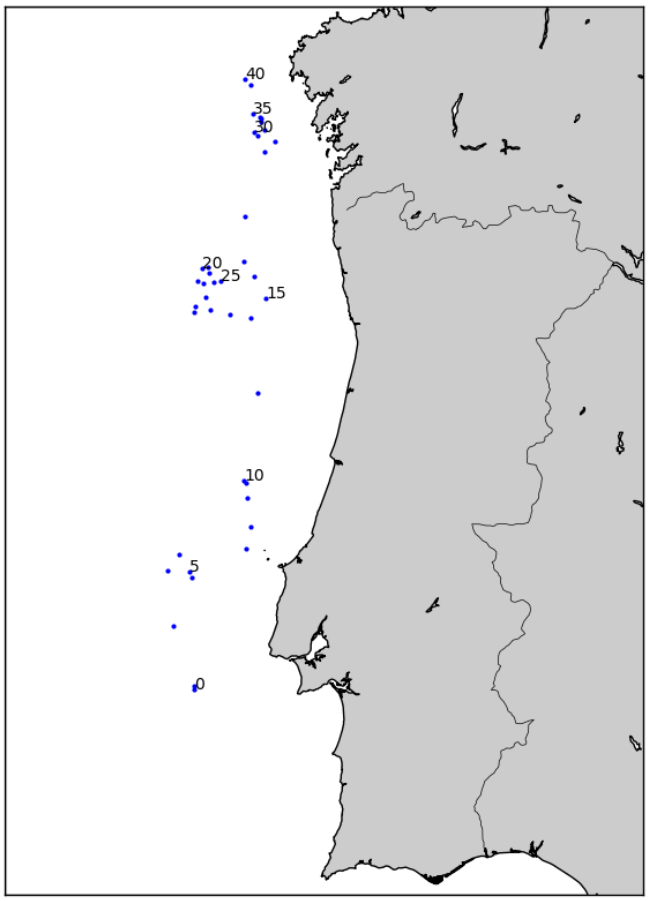


Figura 2.6: Localização dos 40 perfis medidos pela boia WMO-3901909. As coordenadas foram obtidas através do ficheiro referente a esta boia presente nos centros de dados.

2.1.5 Extensões do Projeto Argo

Existem duas extensões do projeto Argo que se encontram em desenvolvimento:

Deep Argo: Este projeto pretende expandir a cobertura vertical da matriz Argo de 2000 metros para 6000 metros de profundidade. Os desenvolvimentos destas boias iniciaram-se em 2014 e atualmente encontram-se 41 boias ativas [11].

BioGeoChemical Argo: Este projeto pretende expandir as variáveis a serem medidas no oceano, através de boias equipadas com bio-sensores para obter medições de variáveis como: oxigénio, nitrato, clorofila e pH do oceano. Encontram-se atualmente 300 boias ativas que contêm estes sensores [12].

[11] https://oceantoday.noaa.gov/deepargo/, visitado a 02,18
[12] http://biogeochemical-argo.org/, visitado a 02/18

2.2 Análise da Bibliografia Argo

A caracterização dos diferentes casos de uso sobre os perfis Argo foi essencial para o trabalho a ser desenvolvido. Devido à grande extensão da bibliografia (mais de 3000 artigos), a escolha e leitura dos artigos resultou ser uma tarefa complicada.

Foi executada uma análise sistemática dos artigos, através do seu conteúdo textual. Foram extraídos: título, resumo, nº citações e palavras chave de cada artigo dos 10 jornais com o maior número de artigos publicados entre 2014 e 2017. Foi também verificado o quartil Scimago [13] para verificar a qualidade das publicações analisadas. A tabela 2.1 apresenta o nome dos jornais/revistas, o número de artigos que foram extraídos e o quartil Scimago e área de estudo associados.

Tabela 2.1: Jornais, número de artigos, e quartil de artigos extraídos que utilizam dados do projeto Argo.

Jornais / Revistas	Nº de artigos	Quartil Scimago (área de estudo)
Journal of Geophysical Research	283	Q1 (Aquatic Science)
Geophysical Research Letters	99	Q1 (Earth and Planetary Sciences)
Journal of Physical Oceanography	90	Q1 (Oceanography)
Climate Dynamics	75	Q1 (Atmospheric Science)
Journal of Climate	71	Q1 (Atmospheric Science)
Ocean Science	42	Q1 (Oceanography)
Ocean Dynamics	40	Q2 (Oceanography)
Deep Sea Research Part I: Oceanographic Research Papers	39	Q1 (Aquatic Science)
Progress in Oceanography	35	Q1 (Aquatic Science)
Ocean Modelling	34	Q1 (Atmospheric Science)

O intuito desta abordagem foi o de agrupar os artigos tendo em conta o seu conteúdo através de métodos de análise textual de forma a conseguir obter uma visão global sobre os diferentes tipos de artigos e consequentemente aplicações dos perfis Argo.

Os dados extraídos também foram utilizados para facilitar a escolha de artigos a ler. Com a informação extraída dos artigos, foi criada uma base de dados relacional de forma a ajudar nas pesquisas realizadas referente às diferentes aplicações. A base de dados contém a informação extraída dos artigos, bem como informação relativa a entidades relevantes dentro do texto. Foi utilizada a API *Google Natural Language*, que contém uma funcionalidade que permite retirar as entidades mais relevantes de texto, para a extração dessas mesmas entidades dos títulos e resumos dos artigos. Cada entidade contém um nome, tipo e, caso exista, um URL de uma página da *Wikipédia* referente à entidade. As figuras 2.8 e 2.9 apresentam o processo descrito para a criação da base de dados e também o esquema de tabelas criado.

[13] https://www.scimagojr.com/journalrank.php

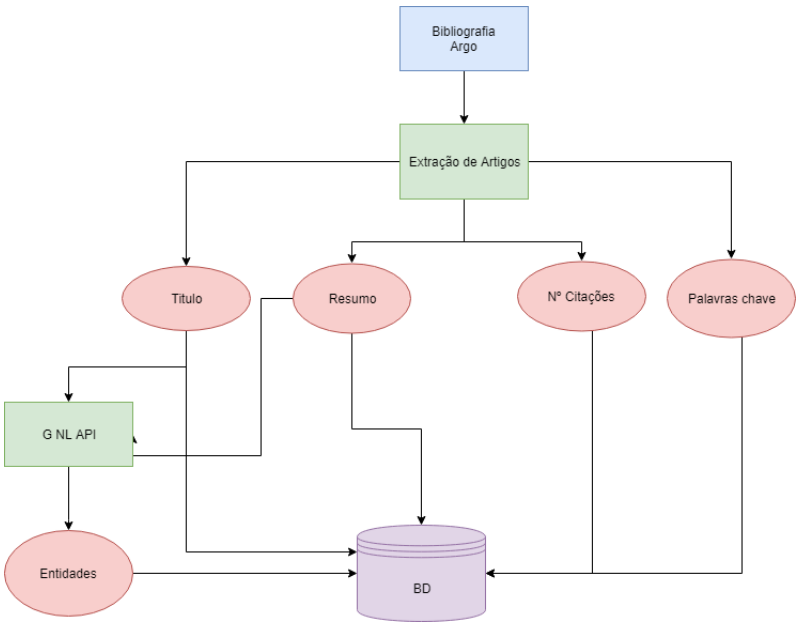


Figura 2.8: Processamento de informação dos artigos contidos na bibliografia Argo.

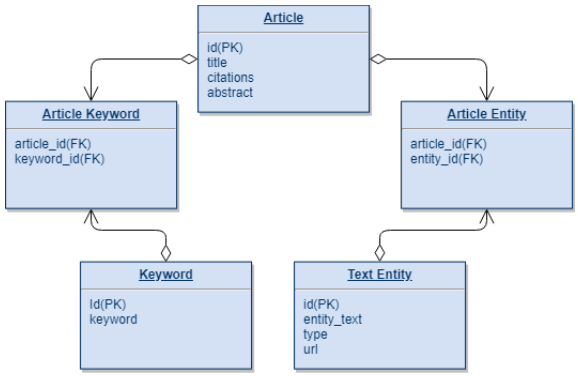


Figura 2.9: Esquema de tabelas para o armazenamento de dados extraídos dos artigos.

A abordagem apresentada contribuiu para a localização de artigos relevantes sobre aplicações dos perfis Argo apresentados na secção 2.3, e ainda, entender as diferentes características relacionadas com os perfis Argo e as suas aplicações. As características espaciais e temporais dos perfis, as diferentes propriedades derivadas que podem ser obtidas (nível do mar, propriedades da camada mista) e diferentes aplicações dos perfis com dados de outras fontes (informação sobre ciclones, dados de satélite), são todos aspetos que foram tidos em conta no processamento e análise destes dados.

A nível prático, em primeiro lugar foram lidos diferentes artigos para obter uma visão geral das aplicações dos dados Argo. A partir dessas aplicações, foram encontrados os artigos na base de dados que referissem certos termos para uma análise mais aprofundada.

O resultado da análise efetuada resultou na caracterização de diferentes aplicações que estão apresentadas na secção seguinte.

2.3 Aplicações de perfis Argo

Os perfis Argo são largamente utilizados em estudos sobre o oceano. Nos 18 anos de atividade deste projeto, já foram efetuados diversos estudos científicos sobre diferentes aspetos do oceano. Esta secção procura mostrar algumas das aplicações que foram levantadas através da análise da bibliografia associada ao projeto Argo.

Algumas aplicações gerais relativas aos perfis incluem:

Variabilidade do oceano: Muitos dos trabalhos efetuados utilizando perfis Argo focam-se no estudo da variabilidade de diferentes propriedades do oceano (temperatura, salinidade, profundidade da camada mista[14],...) a nível regional ou global e a diferentes escalas temporais [5].

Expansão térmica: Através dos perfis argo é possível observar a influência do fenómeno de expansão térmica, devido ao aumento da sua temperatura, no aumento do nível do mar. O uso de perfis Argo juntamente com dados de nível do mar obtidos através de satélites, permite determinar qual é a contribuição da expansão térmica no aumento total do nível do oceano [15] [6] .

ENSO / AMOC: Os dados provenientes do projeto também servem para melhor observar fenómenos naturais como o El Nino Southern Oscilation (ENSO) e Atlantic Meridional Overturning circulation (AMOC) de forma a os melhor compreender e possivelmente prever, devido à sua importância na variabilidade do clima terrestre global [7].

Ciclones e redemoinhos: Os perfis Argo permitem a análise de fenómenos como ciclones e redemoinhos, verificando o seu efeito na sub-superfície do oceano.

De seguida, são apresentadas, de forma mais detalhada, as aplicações específicas dos perfis Argo que demonstram a sua aplicação complementada com dados de diferentes fontes.

2.3.1 Análise de ciclones

Uma das aplicações dos perfis Argo, consiste no estudo do impacto da passagem de ciclones na camada superior do oceano [8]. Neste caso de estudo, analisa-se o impacto de ciclones tropicais na camada mista do oceano, na região Noroeste do oceano Pacífico. A análise é realizada através da localização de perfis Argo situados na trajetória de ciclones, antes e depois da sua passagem.

O estudo em questão foca-se na análise imediata, após a passagem do ciclone, e noutra análise mais tardia para verificar os efeitos da sua passagem a longo prazo. O processo

[14] Camada superior do oceano caracterizada por ter a mesma densidade que a da sua superficie.
[15] https://sealevel.nasa.gov/, visitado em 02/2018

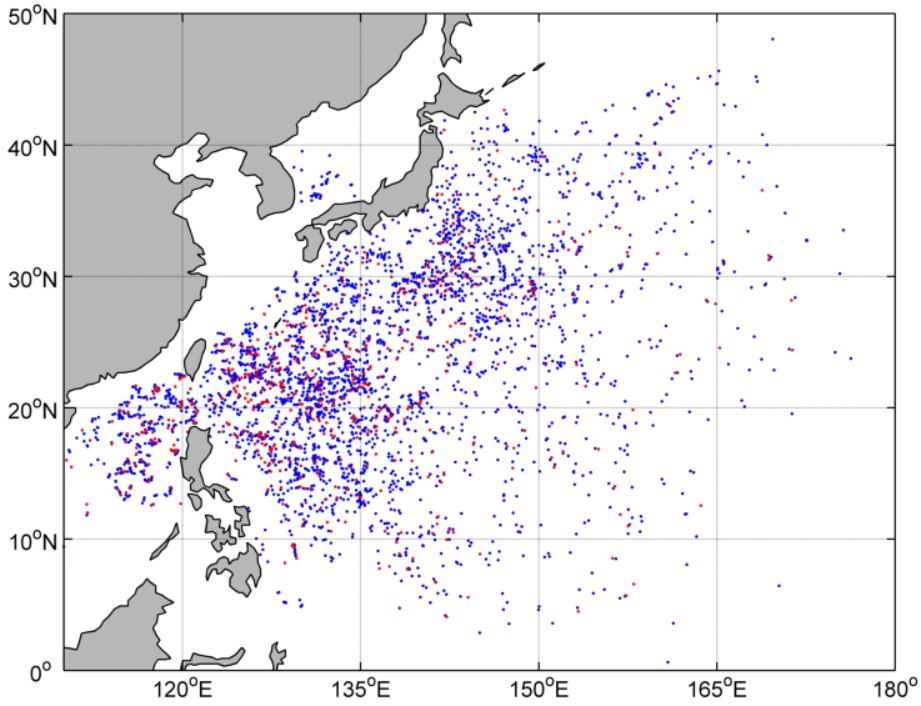


Figura 2.10: Perfis Argo utilizados para a análise da resposta da camada mista na passagem de ciclones: imediata (vermelho) e diferida (azul) [8].

de escolha dos perfis utilizados para a avaliação imediata apresentados na figura 2.10 é efetuada através do seguinte processo:

- São escolhidos os perfis que se encontrem a 500km do centro de um ciclone e no máximo até 6 horas da sua passagem.
- Cada perfil selecionado é emparelhado com um perfil retirado no máximo a uma distância de 50km e 10 dias antes da passagem do ciclone, de forma a analisar o impacto imediato da sua passagem.
- O par definido de perfis é emparelhado com o ciclone correspondente.

Os perfis escolhidos para a análise tardia (pontos a azul na figura 2.10) é semelhante à descrita anteriormente, mas são procurados perfis que se encontrem a 500km do centro de um ciclone e até 10 dias após a sua passagem.

Este caso de uso, demonstra um exemplo de um requisito computacional relacionado com os perfis, ao ser necessária a pesquisa espaciotemporal de perfis Argo, tendo em conta a trajetória de ciclones.

2.3.2 Análise de Redemoinhos (Eddies)

Outro exemplo de aplicação dos perfis Argo é a análise de redemoinhos no oceano. É o caso do estudo [9], que em primeiro lugar procede ao reconhecimento de redemoinhos através de dados de anomalia de nível do mar obtidos através de altimetria por satélite

[16]. Após o seu reconhecimento foi necessário encontrar perfis Argo que se encontrem próximos dos redemoinhos identificados, de forma a poderem ser analisados. O processo descrito encontra-se na figura 2.11. Na parte superior estão apresentados mapas da anomalia do nível do mar (SLA) em diferentes datas, de forma a encontrar possíveis redemoinhos. O redemoinho realçado corresponde ao representado pelas siglas AE (Anticyclonic eddie). Na parte inferior da imagem estão apresentadas o percurso de boias Argo que se cruzaram com o redemoinho na janela temporal contemplada.

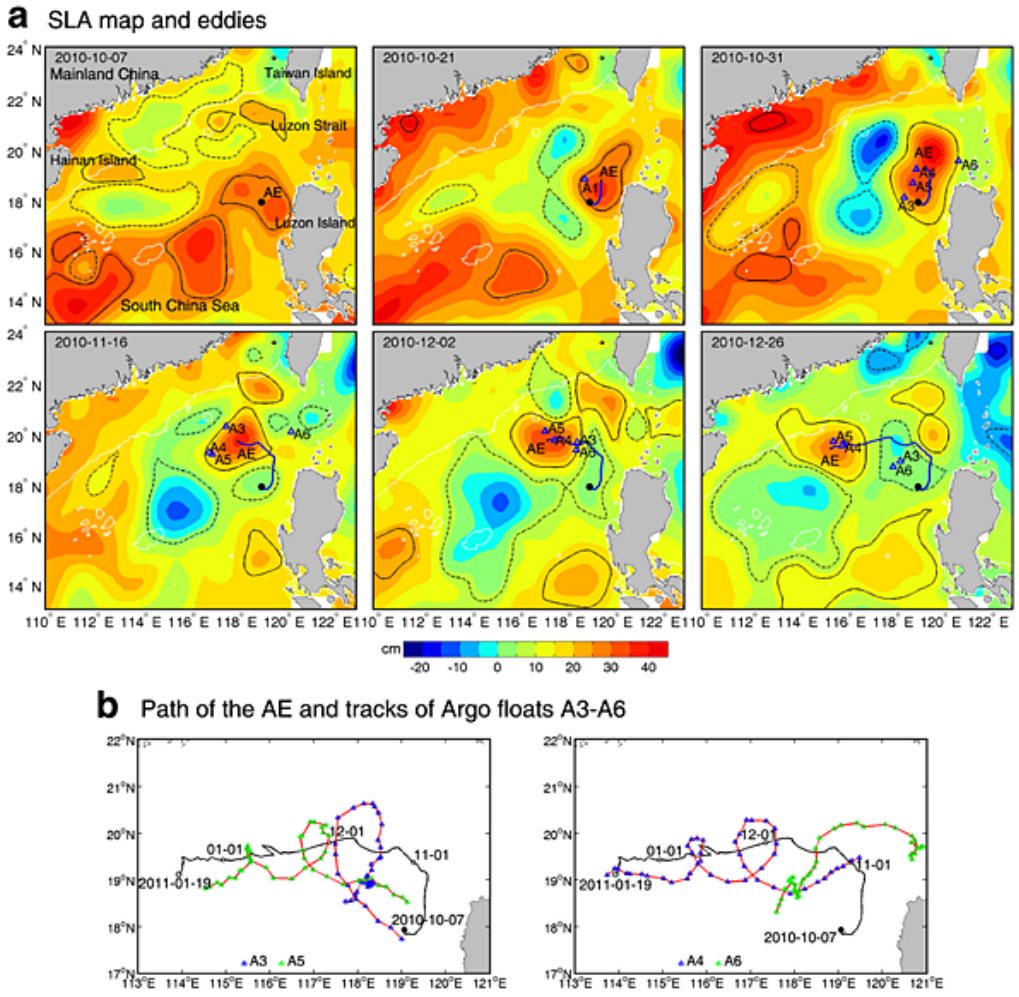


Figura 2.11: Identificação de Redemoinhos e perfis Argo [9].

2.3.3 Produtos Grelha

Uma das dificuldades na análise dos perfis Argo deve-se à sua distribuição temporal e espacial irregular. Existem produtos que através dos perfis criam grelhas com distribuição espacial e temporal uniforme. Estas grelhas permitem uma maior facilidade na análise

[16] https://www.aviso.altimetry.fr/en/data/products/sea-surface-height-products.html, visitado a 02/2018

científica e também possibilitam a agregação de dados de diferentes fontes. Muitos dos artigos referidos anteriormente recorrem a estes produtos nas suas análises.

2.3.3.1 Climatologia Roemmich-Gilson

Um exemplo deste tipo de produtos é a Climatologia de Roemmich-Gilson [5]. Este produto baseia-se apenas em perfis Argo para a criação de uma grelha uniforme com informação sobre a média da temperatura e salinidade entre 2004 a 2016 bem como as anomalias (diferença da média) referentes a cada mês entre os anos referidos.

Este produto apresenta uma resolução espacial de $1° \times 1°$, 58 pontos fixos de pressão, e uma resolução temporal de um mês. Isto significa que são considerados pontos fixos a cada 1º de latitude (-64.5º a 79.5º) e longitude (20.5º a 379.5) e 58 níveis fixos de pressão que correspondem a profundidades entre os 2.5 e 1975 metros. Nestes pontos estão apresentadas tanto as médias de temperatura e salinidade (entre 2004-2016) como as anomalias referentes a cada mês.

Este produto, fornece a temperatura e salinidade média entre 2004 e 2016 para cada um dos níveis de profundidade em cada área definida. Para demonstrar visualmente os dados contidos neste produto, a figura 2.12 apresenta as temperaturas médias num dos níveis de pressão fornecidos (2.5dbar).

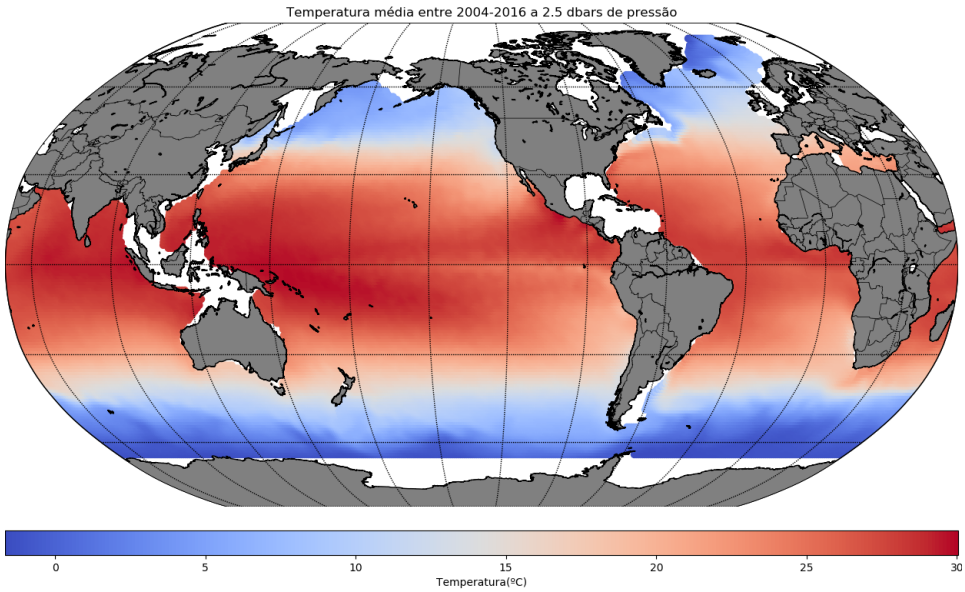


Figura 2.12: Temperatura média entre 2004-2016 a 2.5dbars de pressão [17].

Para além das temperaturas médias, este produto contém ainda as anomalias (diferença da média) mensais de temperatura e salinidade para todos os meses entre 2004-2016. A figura 2.13 apresenta as anomalias referentes a Dezembro de 2016.

[17] Os valores da anomalia de temperatura média foram obtidos em http://sio-argo.ucsd.edu/RG_Climatology.html

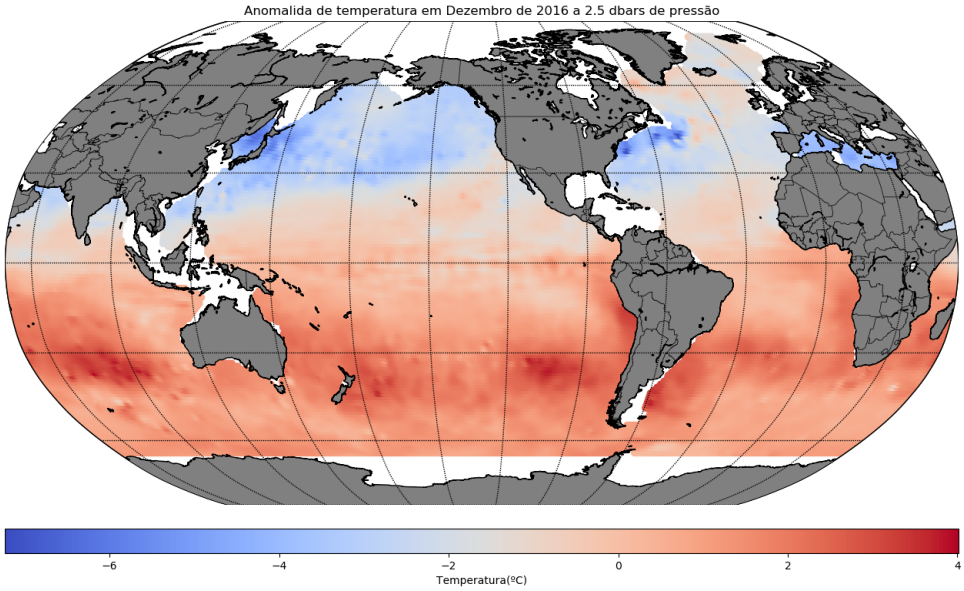


Figura 2.13: Anomalia de temperatura dos oceanos em Dezembro de 2016 a 2.5dbars de pressão.

A criação deste produto envolve em primeiro lugar a interpolação linear dos diferentes perfis Argo, de forma a obter valores de temperatura e salinidade nos níveis fixos de pressão considerados. A média em cada ponto (latitude, longitude, pressão) é depois calculada considerando os 300 pontos mais próximos para cada mês. Dos 300 pontos utilizados, 100 correspondem à pressão do ponto a ser analisado, e os restantes 200 a pontos em profundidades imediatamente próximas. No total, para o cálculo da média em cada ponto, são considerados $3 \times 12 \times 100$ pontos vizinhos, em que 3 corresponde aos diferentes níveis de pressão considerados e 12 aos 12 meses do ano. O processo completo de obtenção da média e das anomalias encontra-se detalhadamente descrito em [5].

2.3.3.2 Outros produtos

Existem também outros produtos com o mesmo propósito mas que derivam outros dados para além da temperatura e salinidade. É o caso do produto YoMaha07 [10], que contém informação sobre correntes marítimas a 1000 metros de profundidade, estimadas através das trajetórias efetuadas pelas boias Argo.

Outro produto deste género, é o *Argo Mixed Layers* [11] que oferece uma base de dados de perfis Argo e uma visão uniforme desses mesmos perfis. A base de dados consiste na informação sobre a camada mista oceânica (profundidade, densidade, temperatura e salinidade) calculada para todos os perfis disponíveis em Fevereiro de 2017, que consiste num total de aproximadamente 1 385 000 perfis. A camada mista é calculada através de dois algoritmos diferentes [12, 13]. Em [12], a profundidade máxima da camada mista oceânica é definida como a primeira profundidade onde a diferença entre a sua temperatura e a temperatura à superfície seja maior que 0.2ºC, ou a diferença de 0.03 kg/m^3

em densidade potencial. O código para o cálculo de todas as propriedades em ambos os algoritmos referidos encontra-se disponibilizado na página do produto [18].

A informação calculada para cada um dos perfis também pode ser acedida sob forma de uma grelha uniforme. Os perfis disponíveis são agregados em divisões de 1º (latitude e longitude) e é calculada a média das propriedades dos perfis que se encontrem localizados em cada divisão. A figura 2.14 apresenta algumas das propriedades presentes na grelha fornecida, mais especificamente: **a)** a profundidade média máxima da camada mista; **b)** o mês onde a profundidade máxima foi observada (se a área conter perfis nos diferentes 12 meses do ano); **c)** o número de perfis considerados em cada área de $1° \times 1°$.

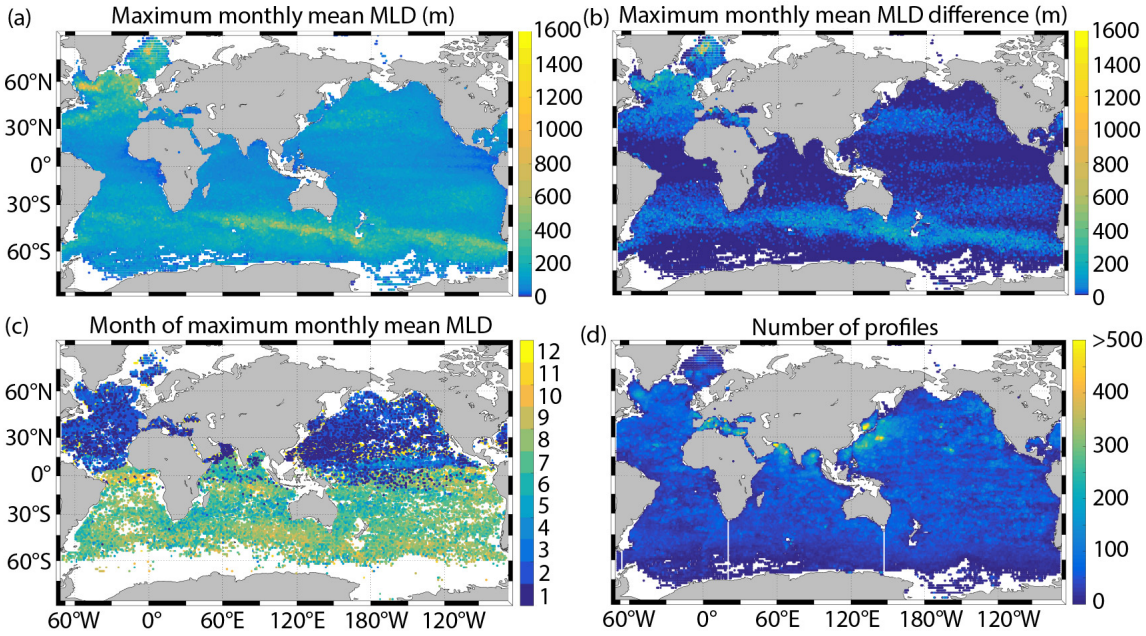


Figura 2.14: Diferentes propriedades *Argo Mixed Layers*[11]

2.4 Conclusões

Neste capítulo foi apresentada uma visão global do projeto Argo, a análise sistemática à sua bibliografia e a descrição das diferentes aplicações associadas aos seus dados. Este trabalho foi realizado com o propósito de apurar as necessidades computacionais de processamento dos dados associados ao projeto.

Na secção 2.3 foram descritas as diferentes aplicações associadas à análise de diferentes fenómenos e variabilidade de diferentes variáveis do oceano. A nível computacional, foi constatado que os perfis Argo são utilizados de duas formas distintas:

Criação de produtos grelha Neste tipo de aplicações os perfis são utilizados para análises a larga escala espacial e temporal. Os perfis são agregados a diferentes pontos

[18] http://mixedlayer.ucsd.edu/, visitado em 02/2018

espaciotemporais (por vezes em conjunto com outras fontes) onde poderão também ser calculadas variáveis derivadas. O produto final corresponde a uma visão uniforme das diferente variáveis consideradas a diferentes granularidades espaciotemporais.

Análise de fenómenos menores Neste tipo de aplicações é necessário encontrar os perfis mais relevantes/próximos espacial e temporalmente do fenómeno, para analisar o seu impacto na camada superior do oceano.

Ambas as formas realçam a importância da componente espaciotemporal dos perfis Argo. Isto motivou a análise do estado da arte de tecnologias capazes de armazenar e processar este tipo de dados. Na secção seguinte, estão apresentadas tecnologias com base em *frameworks Big Data* para o processamento de dados espaciais/espaciotemporais de forma eficiente.

Capítulo 3

Trabalho Relacionado

Neste capítulo é apresentado o trabalho de pesquisa efetuado sobre diferentes *frameworks Big Data* focadas no processamento de dados espaciais e espaciotemporais. As diferentes soluções estudadas têm como base dois diferentes *frameworks* de processamento distribuído: *MapReduce* e *Spark*. Uma pequena introdução das tecnologias fundamentais do ecossistema *Hadoop* encontra-se na secção 3.1.

De seguida, são apresentados alguns algoritmos de particionamento espacial (secção 3.2), que se encontram presentes em alguns dos *frameworks Big Data* referenciados neste capítulo.

Por fim, na secção 3.3 são descritos os *frameworks* encontrados na literatura que expandem *MapReduce* e *Spark* para suportar o processamento de dados espaciais e espaciotemporais.

Está também disponível em anexo o trabalho de pesquisa realizado sobre representação de séries temporais. Na análise das diferentes aplicações de perfis Argo, foi constatado que as séries temporais existentes consistiam na verificação da evolução de variáveis do oceano através de produtos grelha. Este tipo de séries não justificaram o uso prático destas representações.

3.1 Conceitos Fundamentais

O projeto *Hadoop* tem como finalidade o desenvolvimento de tecnologias para computação distribuída de forma escalável e confiável. As tecnologias base deste ecossistema consistem: no sistema distribuído de ficheiros Hadoop Distributed File System (HDFS); no gestor de recursos Yet Another Resource Negotiator (YARN); no *framework* de processamento de dados *MapReduce*.

Nas seguintes subsecções é apresentada uma visão geral de cada uma destas tecnologias.

3.1.1 HDFS

O HDFS é sistema de ficheiros que foi desenhado com o propósito de servir aplicações que necessitem de processar uma grande dimensão de dados. Este sistema foi concebido para ser utilizado em hardware de baixo custo e ser altamente tolerante a falhas.

O *HDFS* segue uma arquitetura *master/worker*. É composta por um *namenode* que contém informação relativa aos diversos ficheiros existentes no sistema, e por vários *datanodes*, onde são guardados os ficheiros. Cada ficheiro é composto por um determinado número de blocos (a ser configurado pelo utilizador), que poderão estar replicados pelos diferentes *datanodes* disponíveis.

Para além dos componentes apresentados, o *HDFS* é caracterizado pelas seguintes propriedades:

- Deteção rápida de falhas e recuperação automática de componentes.
- Modelo *write-once-read-many* para acesso a ficheiros. Após um ficheiro ser criado, escrito e finalmente fechado, as únicas alterações poderão ser do tipo *truncate & append*.
- Possibilidade de movimentar computações para componentes do sistema onde os dados estão localizados, ao invés de movimentar os dados para realizar computações.

Todas estas características promovem altas taxas de transferência no processamento de dados de grande dimensão contidos neste sistema de ficheiros, beneficiando aplicações que o utilizem.

A figura 3.1 apresenta a arquitetura deste sistema de ficheiros. É possível observar um *namenode* que contem os metadados dos ficheiros presentes no sistema, que por sua vez estão repartidos e replicados em blocos pelos diferentes datanodes disponíveis. Para além dos componentes do *HDFS*, estão presentes dois clientes a efetuar operações de leitura e escrita respetivamente.

3.1.2 YARN

YARN é um gestor e alocador de recursos para aplicações executadas em *clusters Hadoop*. Em versões mais antigas do ecossistema *Hadoop* (pré 2.0.0), o *framework MapReduce* tinha a responsabilidade de alocação de recursos no *cluster*, fator que limitava a execução de aplicações fora deste *framework*. O YARN facilitou o desacoplamento entre processamento

[1] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, visitado a 02/19

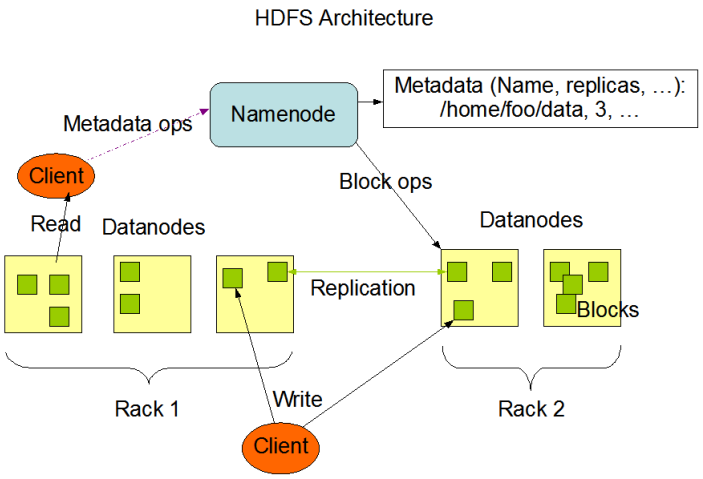


Figura 3.1: Arquitetura do HDFS [1]

de dados e gestão de recursos num *cluster*, algo que permitiu o aparecimento de diferentes frameworks de processamento de dados.

O YARN é composto pelos seguintes componentes:

Resource Manager: Componente responsável pela alocação de recursos no cluster.

Node Manager: Responsável pela execução de tarefas individuais de uma aplicação.

Application master: Componente pertencente ao *framework* de processamento, que gere o estado e necessidade de recursos de uma aplicação.

Container: Unidade de recurso disponível no cluster, incluindo memória e CPU.

Para um *framework* de processamento utilizar o *YARN* como gestor de recursos, este terá que disponibilizar uma implementação do *Application Master*, onde é especificada de que forma os recursos do *cluster* subjacente são utilizados.

Na figura 3.2, é possível observar um exemplo contendo duas aplicações em execução simultânea num *cluster* (representadas pelas duas cores). Cada aplicação contém um *application master* que requisita recursos disponíveis ao *ResourceManager*, sendo estes disponibilizados sobre a forma de *containers* nos diferentes *NodeManagers*.

3.1.3 MapReduce

MapReduce [14] é um *framework* para execução de programas em ambientes distribuídos. O modelo de programação deste *framework* é expresso por duas funções:

- **map:** Esta função, recebe todos os elementos do conjunto de dados providenciado pelo utilizador e gera um conjunto intermédio de pares chave-valor.

[2] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, visitado a 02/19

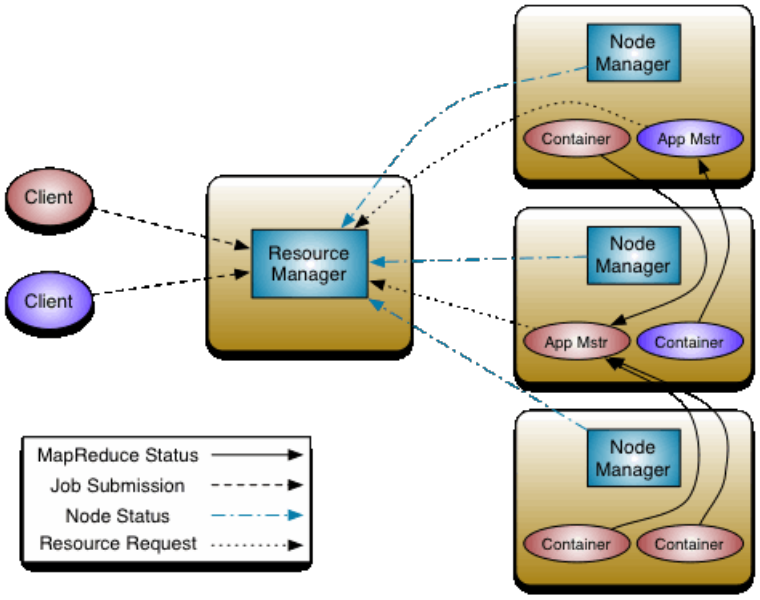


Figura 3.2: Arquitetura do YARN [2]

- **reduce:** Esta função, recebe todos os pares intermédios chave-valor com a mesma chave para gerar um resultado final.

Programas expressos por estas funções são altamente paralelizáveis em ambientes distribuídos. A figura 3.3 apresenta o fluxo de execução de um programa *MapReduce*, composto pelos seguintes passos:

- 1 - Os ficheiros de input são divididos em M partes (cada parte geralmente corresponde a um bloco do HDFS) e várias cópias do programa são executadas nos diferentes nós do *cluster*.
- 2 - O programa mestre (*JobTracker*) distribui as diferentes tarefas *map* e *reduce* pelos nós (*TaskTracker*) que se encontrem disponíveis.
- (3,4) - Cada nó que tenha uma tarefa map atribuída, lê a divisão do input original respetiva e executa a função *map* definida pelo programador, sendo o seu resultado guardado em memória. Periodicamente, os pares resultantes da função *map* em memória, são escritos em disco, sendo o nó mestre informado da sua localização.
- (5,6) - Quando um nó responsável por uma tarefa *reduce* é informado da localização de resultados intermédios de uma função *map* em disco pelo nó mestre, este transfere-os e organiza-os por chave. Após organizados, a função *reduce* definida pelo programador é executada e os seus resultados são escritos num ficheiro.

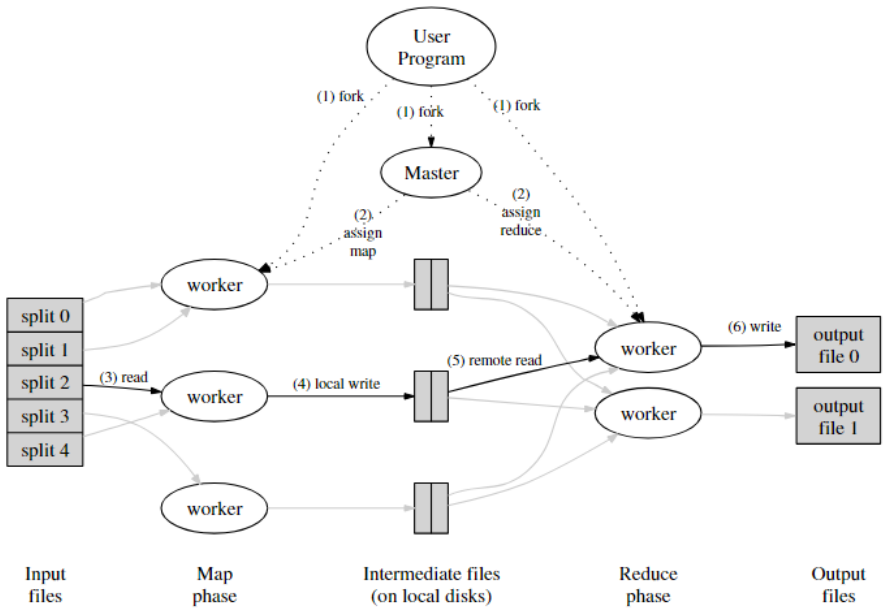


Figura 3.3: Fluxo de execução de um programa MapReduce, retirado de [14].

3.1.4 Spark

Apache Spark é um *framework* de computação distribuída, com o foco principal em processamento de dados em memória. A principal abstração deste *framework* são os Resilient Distributed Dataset (RDD) [15], estruturas de dados imutáveis, particionadas por diferentes nós de um *cluster*, que permitem a execução de diferentes operações sobre dados, de forma distribuída e tolerante a falhas.

Uma aplicação *Spark* segue uma arquitetura *master-worker* e é composta por um *driver* e diferentes *executors*. O *driver* é um programa mestre responsável pela declaração das operações sobre RDD e pela sua divisão em diferentes tarefas, enquanto que os *executors* são os responsáveis pela execução dessas mesmas tarefas e pela persistência de RDDs em memória ou em disco. A figura 3.4 apresenta um exemplo dos diferentes componentes quando é executada uma aplicação *Spark* num *cluster*.

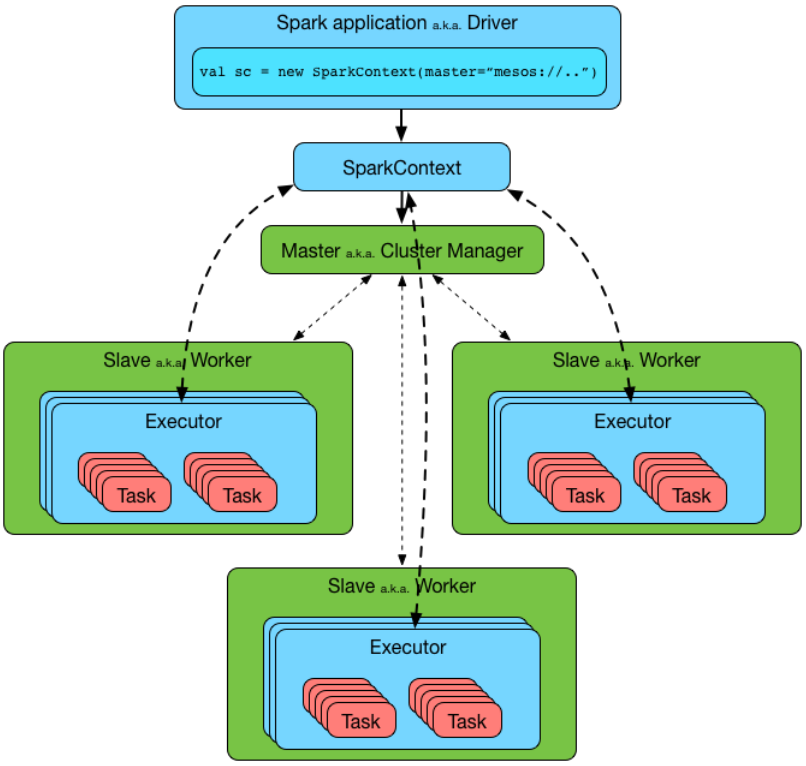


Figura 3.4: Fluxo de execução de um programa *Spark*[3].

3.2 Técnicas de particionamento espacial

Alguns *frameworks* apresentadas na secção 3.3 possibilitam a criação de índices que dividem eficientemente dados tendo em conta a sua componente espacial / espaciotemporal. Estes índices são utilizados para obter um melhor desempenho na execução de pesquisas com predicados espaciais. Mais concretamente, os diferentes *frameworks* têm como foco uma ou mais das seguintes pesquisas:

Range Query Devolve os resultados encontrados dentro de um determinado intervalo espacial.

KNN Query Dado um ponto (x,y) e um valor K, são encontrados os K vizinhos mais próximos do ponto.

Join Query Dados dois conjuntos de dados indexados X e Y e um predicado A, encontra todos os pares (X_i,Y_i) que verifiquem A.

Esta secção apresenta abreviadamente alguns dos métodos encontrados.

3.2.1 Quadtree

Quadtrees [16] são árvores em que cada nó interno contém exatamente quatro nós filhos, que dividem o espaço bidimensional em quatro quadrantes. Cada nó neste tipo de árvore

é caracterizado por uma área restrita e por um conjunto de elementos. Se o conjunto de elementos de um nó ultrapassar um limite definido pelo programador, este divide a sua área representativa em quatro quadrantes, e cada quadrante passa a ser representado por um filho desse nó e os seus elementos são distribuídos.

Na figura 3.5 é possível observar um exemplo de divisão espacial de dados numa *quad-tree*. No lado esquerdo da imagem é apresentada a distribuição de pontos bidimensionais. No lado direito da imagem está apresentada a distribuição das diferentes partições na árvore.

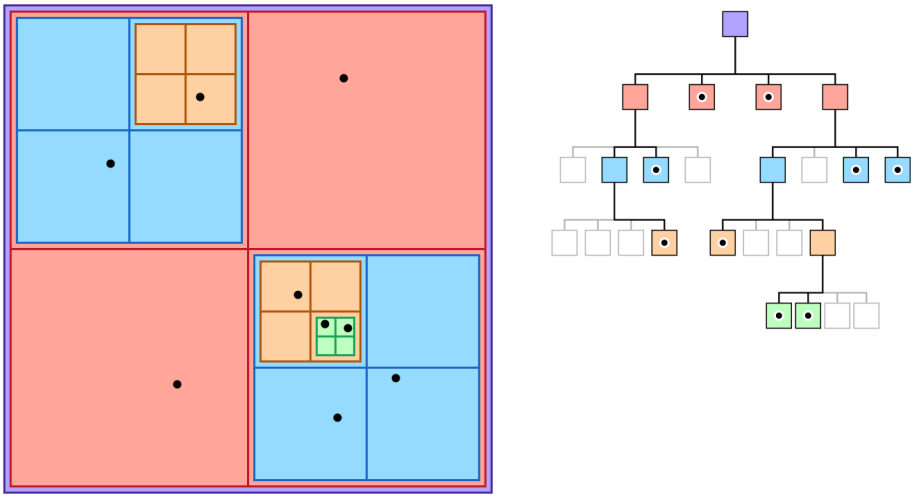


Figura 3.5: Exemplo de Quadtree [4].

3.2.2 Rtree

Uma R-tree [17] é uma árvore equilibrada em que cada nó não-folha representa uma área rectangular que engloba totalmente os seus nós filhos, sendo que as folhas contêm os elementos espaciais a serem indexados. A ideia geral consiste em agrupar elementos espaciais próximos pelos seus MBR.

Existem dois aspetos importantes a referir na criação destas árvores que impactam a sua estrutura final. Um dos aspetos refere-se à escolha de sub-árvores na inserção de um novo elemento na árvore. Um exemplo de heurística utilizada consiste em inserir o novo elemento na sub-árvore que necessite de menor alargamento espacial. Outro aspeto a ter em conta é a necessidade da divisão das folhas quando estas ultrapassam a sua capacidade máxima, onde existem diferentes formas de dividir os elementos espaciais contidos na folha em dois novos nós.

A figura 3.6 apresenta um exemplo de uma *R-Tree*, onde é possível constatar que os nós superiores contêm na totalidade todos os nós inferiores relacionados e os objetos espaciais estão contidos nas suas folhas.

[4] https://developer.apple.com/documentation/gameplaykit/gkquadtree, visitado a 02/2019
[5] https://developer.apple.com/documentation/gameplaykit/gkrtree, visitado a 02/2019

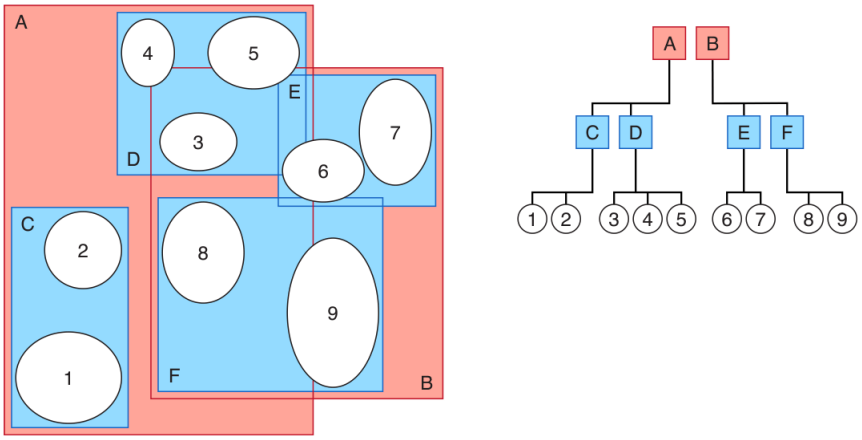


Figura 3.6: Exemplo de Rtree [5].

3.2.3 Kdtree

Uma KdTree [18] é uma árvore binária em que cada nó é constituído por um ponto K-dimensional, sendo K o número de dimensões presentes nos dados a organizar. Cada nó não-folha pode ser considerado como a divisão binária do espaço de uma das dimensões existentes. Em cada nível da árvore é apenas considerada uma das dimensões para a divisão espacial.

Um exemplo de uma K-d tree de 2 dimensões pode ser observada na figura 3.7, onde é feita uma subdivisão do plano X e Y alternadamente entre cada nível da árvore.

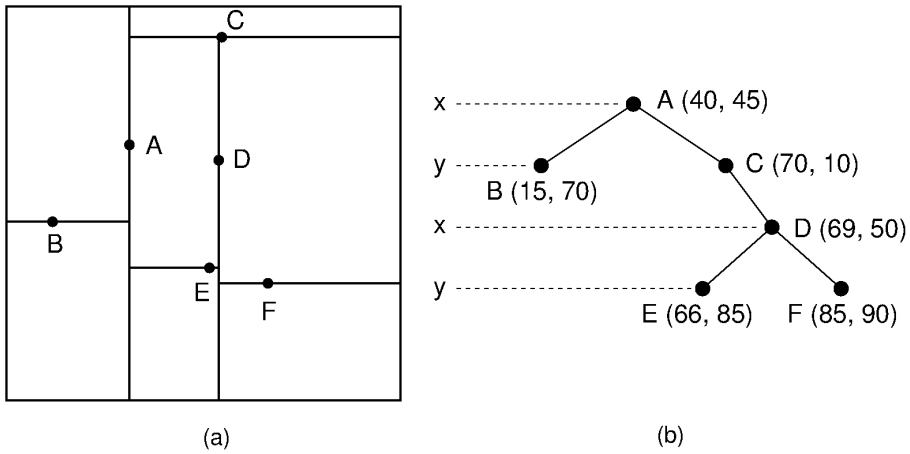


Figura 3.7: exemplo de uma K-d Tree.

3.2.4 Z-order / Hilbert curves

Z-order e Hilbert são tipos de curvas de preenchimento de espaço que possibilitam o mapeamento de dados multi-dimensionais numa única dimensão, mantendo a sua localidade. Esta característica é adequada para o particionamento de dados espaciais, visto ser possível a criação de uma curva que englobe a totalidade do espaço dos dados e transformar a

sua representação espacial numa única dimensão, sendo que valores próximos na curva correspondem a dados espacialmente próximos.

Um exemplo é apresentado a figura 3.8, onde é demonstrado um exemplo do mapeamento do espaço bidimensional numa curva *Hilbert*, onde é possível observar que os pontos espacialmente próximos têm representações próximas na curva.

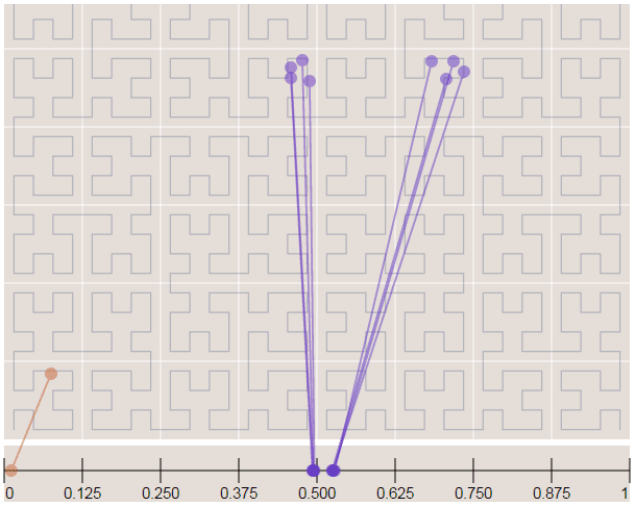


Figura 3.8: Mapeamento de uma Hilbert curve para o espaço bidimensional [6].

Embora o principio seja o mesmo, as curvas *Hilbert & Z-order* diferem na forma como mapeiam o espaço multidimensional. Na figura 3.9 é possivel observar como as duas curvas mapeiam o mesmo espaço bidimensional, onde também é possível constatar que no mapeamento da z-order existem alguns "saltos"na curva, apresentando pior localidade dos dados do que a curva *Hilbert*

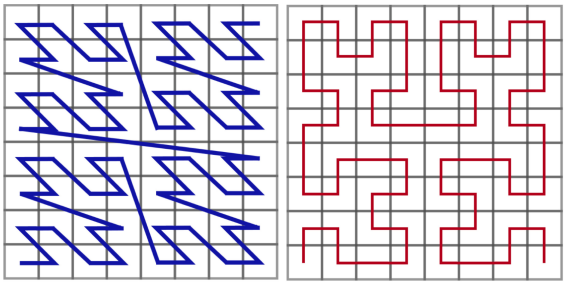


Figura 3.9: Z-order vs Hilbert

3.3 Big Spatial/Spatio-temporal Data

O crescimento sistemático do número de perfis produzidos pelo projeto Argo, o desenvolvimento das suas extensões (Deep Argo e BGC-Argo) que pretendem aumentar a cobertura a nível espacial e o número de variáveis a serem medidas, e os diferentes casos de uso

[6] http://bit-player.org/extras/hilbert/hilbert-mapping.html, visitado a 02/2019

que necessitam de fontes de dados complementares, motivaram a pesquisa de tecnologias Big Data para o processamento e análise destes dados. Devido à componente espacial e temporal associada aos perfis Argo, foi estudado o estado da arte em tecnologias que tratam dados com estas características.

A pesquisa sobre este tema foi inicialmente baseada num estudo [19] que apresenta o estado da arte em relação ao uso de tecnologias *Big Data* para o processamento de dados com componente espacial, sendo que algumas também têm em conta a sua possível componente temporal.

O estudo divide as tecnologias estudadas em 6 aspetos, que foram utilizados como base para a caracterização das tecnologias encontradas:

Implementação As tecnologias de processamento deste tipo de dados poderão ter três abordagens de implementação:

- A componente espacial é definida sobre tecnologias já existentes *(on-top)*
- A tecnologia tem módulos disponíveis para o tratamento de dados espaciais *(built-in)*
- A tecnologia é construída com o foco no tratamento de dados espaciais. *(From-scratch)*

Estas abordagens de implementação apresentam diferentes vantagens e desvantagens. Uma implementação *on-top* é a mais simples, já que a tecnologia utilizada é apenas vista como uma caixa-preta. Mas em contrapartida, este tipo de implementação não está otimizada para o tratamento de dados com componente espacial. Portanto, esta abordagem embora seja a mais simples, à partida será a de pior desempenho entre as restantes.

A abordagem *from-scratch* é especializada para dados espaciais, o que leva a que tenha melhor desempenho, mas o foco único nestes dados significa que para operações envolvendo dados não espaciais poderá ser pior que as restantes abordagens.

Implementações *built-in* juntam as vantagens das anteriores ao apresentarem soluções para tipos de dados espaciais e não espaciais. O problema deste tipo de implementação, encontra-se no alto acoplamento dos componentes a uma versão específica de um sistema base. A atualização do sistema base poderá quebrar determinadas funcionalidades neste tipo de soluções, sendo a única solução a sua atualização para a versão mais recente do sistema.

Arquitetura Identifica qual a arquitetura subjacente ao sistema.

Linguagem Verifica a existência de uma linguagem de alto nível para interação com o sistema.

Índice Espacial Outro dos parâmetros de avaliação e comparação das diferentes tecnologia foi a verificação da existência de índices espaciais. Este tipo de índice tem em

conta as propriedades espaciais dos dados e o seu particionamento pelos diferentes nós do sistema distribuído, é feito tendo em conta a localização dos dados. A ideia consiste em agrupar dados espacialmente próximos nas mesmas partições de forma a melhorar o desempenho de possíveis pesquisas que contenham predicados espaciais/espaciotemporais sobre esses dados.

Pesquisas Especifica quais os tipos de pesquisas suportados pelo sistema.

Os diferentes aspetos apenas têm em conta a componente espacial dos dados, embora os critérios utilizados possam ser adaptados para dados espaciotemporais. Nessa ordem de ideias, foram analisados estudos sobre tecnologias *Big Data* com o foco em dados espaciotemporais. Para a análise de cada uma, recorreu-se aos critérios supra-referidos.

As soluções encontradas caracterizam-se pela possibilidade de expansão das funcionalidades de diferentes tecnologias baseadas no ecossistema Hadoop[7] para suportar pesquisas espaciais / espaciotemporais de forma eficiente. De seguida, serão apresentadas brevemente algumas arquiteturas estudadas que permitem o processamento e pesquisas sobre dados com componente espacial e/ou temporal.

3.3.1 CloST

O CloST [20] propõe uma solução baseada em Hadoop (MapReduce) de forma a otimizar pesquisas com predicados espaciais e temporais. É definido um modelo em que os dados são guardados sob a forma de tabelas contendo os seguintes atributos: **(Oid,Loc,Time,A1 ,...,An)** em que **Oid** corresponde ao **Id** de um objeto, **Loc** à sua localização, **Time** ao tempo relacionado, e **A1...An** a outros atributos.

As pesquisas suportadas consistem na procura de um certo elemento, ou todos os elementos que se encontrem a determinado intervalo espacial e temporal definidas como: **Q(I,S,T)** e **Q(S,T)**. *S* e *T* correspondem a intervalos espaciais e temporais, e *I* corresponde ao identificador de um certo objeto.

De forma a otimizar estas pesquisas, os dados originais são particionados hierarquicamente em 3 níveis distintos:

1. Os dados são primeiro divididos em diferentes partições (nível 1) tendo em conta os diferentes valores de Oid e intervalos de tempo de baixa granularidade.

2. Cada partição nível-1 é depois dividida em partições (nível-2) de acordo com o atributo espacial (Loc) de cada elemento.

3. Finalmente, cada partição nível-2 é ainda dividida em partições (nível-3) tendo em conta intervalos de tempo com maior granularidade.

[7] http://hadoop.apache.org/, visitado a 02/2018

As partições de nível 1/2 servem como índices para encontrar elementos que estão localizados nas partições nível-3. Todas as partições (nível 1-3) são guardadas como ficheiros no HDFS.

3.3.2 Array climate data

Em [21] é proposta uma solução baseada em MapReduce para pesquisas espaço-temporais sobre dados climáticos vetoriais (semelhantes aos produtos grelha referidos na secção 2.3.3) caracterizados pelas dimensões de latitude, longitude, altitude e tempo. Os dados originais são repartidos em grelhas de duas dimensões e é criado um índice espacio-temporal relacionando a informação temporal e espacial das diferentes grelhas extraídas dos dados originais com o local onde se encontram guardadas. Cada grelha de 2 dimensões (latitude e longitude) contém medidas de uma variável a um determinado tempo e altitude: **G(V,T,A)**

É suportada a pesquisa **STQuery(Var,Cov,Alt,Time)** para a pesquisa das grelhas, onde:

Var Corresponde à variável a ser pesquisada.

Cov Área de cobertura desejada expressa por valores de latitude e longitude.

Alt Subconjunto de valores de altitude a serem considerados.

Time Subconjunto de valores temporais a serem considerados.

Esta pesquisa devolve um subconjunto dos dados originais em diferentes grelhas de duas dimensões.

3.3.3 SpatialHadoop

SpatialHadoop [22] é um *framework Hadoop (Mapreduce)* com objetivo de providenciar suporte nativo para dados espaciais. Este *framework* permite a criação de índices com a finalidade de serem utilizados em programas *MapReduce* de forma a otimizar o desempenho de operações espaciais.

Os índices espaciais são criados através da divisão do conjunto de dados original em várias partições, utilizando diferentes algoritmos de particionamento espacial (especificados na secção 3.2). O conjunto de dados indexado resultante, consiste em vários ficheiros, onde cada um contém os dados relativos a cada partição espacial e num ficheiro mestre com a informação relativa aos limites espaciais de cada partição física. Programas *MapReduce* contendo predicados espaciais beneficiam desta estrutura, já que o ficheiro mestre do índice espacial pode ser consultado para saber quais as partições necessárias para a computação.

São disponibilizados de raiz programas *MapReduce* correspondentes às seguintes pesquisas espaciais:

Range Query Devolve os resultados encontrados dentro de um determinado intervalo espacial.

KNN Query Dado um ponto (x,y) e um valor K, são encontrados os K vizinhos mais próximos do ponto.

Join Query Dados dois conjuntos de dados indexados X e Y e um predicado A, encontra todos os pares (X_i,Y_i) que verifiquem A.

É também disponibilizada uma linguagem de alto nivel denominada de Pigeon [23], que adiciona tipos de dados (SPOINT) e predicados espaciais (DISTANCE, OVERLAPS) à linguagem de scripting *Pig Latin*[8]. *Pig Latin* é uma linguagem de scripting de alto nível que é compilada para uma sequência de programas *MapReduce* de forma eficiente.

3.3.4 ST-Hadoop

ST-Hadoop [24] é uma extensão do *framework SpatialHadoop* com o objetivo de providenciar suporte nativo para armazenamento e pesquisa de dados espaciotemporais. Este *framework* suporta a criação de um índice espaciotemporal replicado a diferentes granularidades temporais, de forma a permitir o melhor desempenho para pesquisas com diferentes intervalos temporais.

Para a formulação de pesquisas, é utilizada uma extensão da linguagem Pigeon [23], que adiciona suporte a tipos de dados temporais (STPoint, TIME e INTERVAL) e também a diferentes predicados (OVERLAP e JOIN). A figura 3.10 apresenta um exemplo de uma pesquisa espaciotemporal utilizando esta linguagem.

```
Objects = LOAD 'points' AS (id:int, STPoint:(Location,Time));
Result  = FILTER Objects BY
          Overlaps (STPoint, Rectangle(x1, y1, x2, y2), Interval (t1, t2) );
```

Figura 3.10: Pesquisa por intervalo de tempo e espaço [24].

São suportados dois tipos diferentes de pesquisas espaciotemporais:

ST-Range Query Esta pesquisa é especificada por dois predicados, uma área espacial e um intervalo temporal, A e T respetivamente e encontra um conjunto de elementos que se encontrem dentro da região A e no intervalo T.

ST Join Dados dois *datasets*(R e S) contendo elementos ST devidamente indexados, e um predicado ST P, a operação retorna todos os pares de elementos <r,s> que satisfaçam P.

[8] https://pig.apache.org/

3.3.5 Geomesa

Geomesa [25] é um sistema que utiliza como base, a estrutura de armazenamento de dados Chave-Valor distribuída Accumulo[9] possibilitando o armazenamento de dados espaço-temporais.

De forma a indexar os dados tendo em conta a sua componente temporal e espacial, são utilizadas *Z-order curves* (apresentadas na secção 3.2).

Os valores são armazenados na base de dados, utilizando uma chave que contém informação espaciotemporal do objeto a ser introduzido. A figura 3.11 apresenta a estrutura de chave-valor utilizada, em que Z3 corresponde ao resultado de uma *Z order curve* com três dimensões (Latitude, Longitude, Tempo).

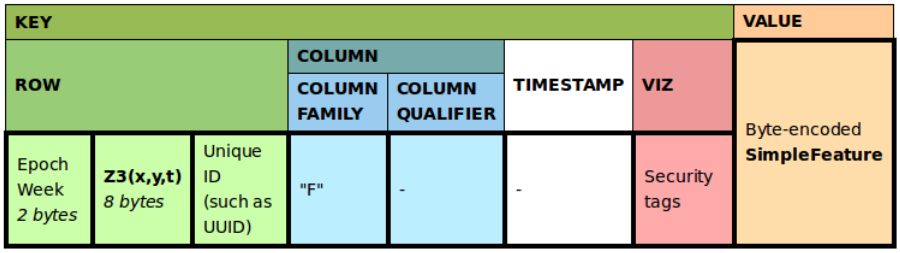


Figura 3.11: Estrutura Chave-Valor utilizada para o armazenamento de dados[10].

É suportada uma linguagem denominada de ECQL (Extended Common Query Language), definida pela OGC (Open Geospatial Consortium)[11]. A linguagem ECQL fornece operadores espaciais para suportar pesquisas contendo intervalos ou interseções espaciais. Esta linguagem possui também diversos operadores temporais, que incluem expressões como: **BEFORE**, **AFTER**, **DURING**.

3.3.6 Geospark

Geospark [26] é um *framework* baseado em *Apache Spark* para o processamento de dados espaciais em larga escala. Este *framework* introduz uma extensão do RDD designada de SRDD com as seguintes particularidades:

- Facilidade de criação de SRDD a partir de diferentes fontes de dados espaciais (ex: criação de um SRDD diretamente de um ficheiro csv contendo tipos de dados espaciais).
- Implementação eficiente e paralela de diferentes tipos de *queries* espaciais: *Range*, *KNN e Join queries*.
- Possibilidade de criação de índices, utilizando *R-trees* ou *Quadtrees*, com o intuito de otimizar as pesquisas espaciais.

[9] https://accumulo.apache.org/, visitado a 02/2018
[11] http://www.opengeospatial.org/, visitado em 02/2018

A introdução desta abstração facilita a criação de aplicações de processamento de dados espaciais de forma eficiente, em *Apache Spark*.

O Geospark também disponibiliza um serializador personalizado, que permite guardar os SRDD de forma eficiente na memória de um *cluster*, durante a execução de um determinado programa.

3.4 Conclusões

Neste capítulo foi apresentado todo o trabalho relacionado realizado nesta dissertação. Foram apresentados conceitos fundamentais de tecnologias *Big Data*, técnicas de particionamento espacial e o estado da arte de tecnologias *Big Data* para o tratamento de dados espaciais / espaciotemporais.

De todos os *frameworks* estudados neste capítulo, são de realçar: *SpatialHadoop*, *ST-Hadoop* e *Geospark*. Estes *frameworks* consistem em soluções *built-in* sobre *MapReduce* e *Spark* e apresentam todos os critérios referidos na secção 3.3 (apresentados na tabela 3.1):

Tabela 3.1: Frameworks a avaliar.

	SpatialHadoop	ST-Hadoop	Geospark
Linguagem	Pigeon	Pigeon	SparkSQL
Arquitetura	Built-in	Built-in	Built-in
Índices	Espaciais	Espaciotemporais	Espaciais
Pesquisas	Range / KNN	Range	Range / KNN
Framework	MapReduce	MapReduce	Spark

- Linguagem de alto nível para o processamento de dados espaciais (Tipos e operações).
- Arquitetura Built-In nos *frameworks* MapReduce e Spark.
- Capacidade de criar índices com diferentes algoritmos de particionamento espacial / espaciotemporal.
- Implementação de pesquisas Range e KNN eficientes, utilizando os índices criados.

Para além destes pontos fundamentais, todos os *frameworks* referidos são *open-source* e possuem boa documentação sobre as suas funcionalidades. Estes aspetos foram importantes na aprendizagem e implementação de operações sobre estes *frameworks*, no processamento de perfis Argo e no desenvolvimento de um produto grelha.

O conteúdo restante da dissertação foca-se no uso prático destas tecnologias, e como otimizar o processo de criação de índices e Pesquisas espaciais / espaciotemporais.

Capítulo 4

Análise Experimental

Após a escolha dos *frameworks Big Data* e a configuração da infraestrutura a utilizar, foi desenhado um plano experimental para avaliar o desempenho destes *frameworks* no processamento de perfis Argo.

Neste capítulo é apresentado o plano experimental desenhado, bem como os resultados experimentais obtidos na avaliação dos diferentes frameworks avaliados.

4.1 Introdução

Uma visão global do plano experimental está sintetizada na figura 4.1 e está divida em quatro partes distintas:

Input: Pré processamento e introdução dos dados Argo no cluster.

Framework: Introdução dos dados pré processados nos diferentes *frameworks* estudadas.

Operações: Operações realizadas utilizando cada *framework*, que se dividem em duas categorias: Criação de Índices e *Queries*.

Resultados: Extração de indicadores das operações realizadas.

A primeira etapa do processo experimental consistiu na obtenção e pré-processamento dos dados de perfis Argo, onde foi extraída informação (descrita na secção 4.2) relativa a todos os perfis entre 1997 e 2018, resultando num único ficheiro de 61GB contendo 683 763 237 medições de perfis.

De seguida foi criado um conjunto de *queries* com predicados espaciais e espacio-temporais com o objetivo de avaliar o desempenho das diferentes *frameworks*. As *queries*

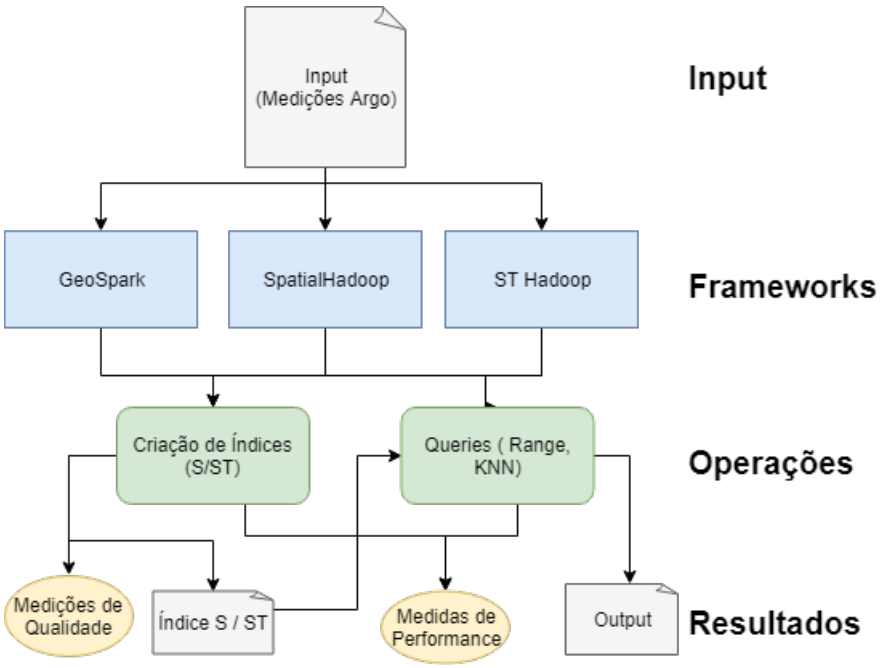


Figura 4.1: Arquitetura experimental.

foram parametrizadas tendo em conta a distribuição não uniforme dos perfis Argo e foram escolhidas duas zonas distintas com diferente densidade de dados. Este processo está descrito na secção 4.3.

Em cada *framework* selecionada para avaliação (SpatialHadoop, ST-Hadoop, GeoSpark), foram analisados dois tipos diferentes de operações:

Criação de índices: Esta operação consiste na criação de índices espaciais/espaciotemporais a partir das medições dos perfis Argo extraídos, onde foi avaliado o seu desempenho e escalabilidade. Nos frameworks SpatialHadoop e ST-Hadoop também foi possível retirar indicadores associados à qualidade do índice criado.

Queries: As *queries* escolhidas para avaliar as *frameworks* foram executadas sobre os diferentes índices criados permitindo avaliar o seu desempenho e escalabilidade. No caso do *framework* GeoSpark, as *queries* foram efetuadas sobre o conjunto original de dados.

Por fim, na secção 4.7 são apresentadas as conclusões finais obtidas neste trabalho experimental.

4.2 Pré-processamento de perfis

Os dados utilizados na avaliação experimental, correspondem a todos os perfis de boias Argo existentes no GDAC a 15 de Março de 2018. Os dados dos perfis foram alvo de uma fase de pré-processamento que consistiu na extração de informação individual de cada

perfil, através da biblioteca Java NetCDF[1] para a leitura e extração de dados em formato NetCDF, e colocada num único ficheiro. Foram apenas extraídas medições de perfis que contivessem valores de posição e data válidos.

Cada linha do ficheiro processado contém uma medição de um perfil que é caracterizado pelas variáveis apresentadas na tabela 4.1. Após o pré-processamento dos perfis, o conjunto de dados final contém 683.763.237 medições. O ficheiro foi armazenado no HDFS presente em cada grupo de infraestrutura utilizado.

Tabela 4.1: Variáveis extraídas de cada perfil Argo.

Variável	Descrição
Identificador	Identificador associado ao perfil
modo	Modo do perfil. "R"para perfis em tempo real e "D"para modo diferido
data	Data de medição de um perfil
latitude	Latitude do local de medição do perfil
longitude	Longitude do local de medição do perfil
temperatura	Valor de temperatura de uma medição de um perfil
salinidade	Valor de salinidade de uma medição de um perfil
pressão	Valor de pressão de uma medição de um perfil
erro_pressão	erro associado ao valor de pressão
erro_salinidade	erro associado ao valor de salinidade
erro_temperatura	erro associado ao valor de temperatura

A opção de pré-processar e armazenar a informação dos perfis pelas suas medições deve-se ao facto de ser necessário responder a questões que necessitam do processamento das medições de cada perfil, ao ínves do perfil apenas. Este aspeto foi importante na definição de *queries* de avaliação, já que será apenas relevante analisar questões que necessitem a análise das medições de cada perfil. Os *frameworks* em análise permitem apenas a indexação de duas coordenadas espaciais. Uma medição Argo corresponde a um ponto tridimensional contendo as variáveis (latitude, longitude, pressão) mas para a indexação foram consideradas apenas as duas primeiras.

4.3 Descrição e parametrização de Queries

O conjunto de *queries* implementado para avaliar o desempenho das diferentes *frameworks* foi o seguinte:

Q1: Dado um intervalo espacial / espaciotemporal, devolver o número de medições que nele se encontram.

Q2: Dado um intervalo espacial / espaciotemporal, retornar todos os pares (P_i, M_i), onde P_i representa o identificador do perfil e M_i o número de medições que este contém.

[1] https://www.unidata.ucar.edu/software/thredds/current/netcdf-java/

Q3: Calcular a média da temperatura de cada perfil numa determinada área espacial / espaciotemporal.

Q4: Encontrar as K medições mais próximas de um determinado ponto.

Este conjunto de *queries* foi escolhido para avaliar especificamente as operações *range* e *KNN* de cada *framework* tendo em conta as medições de cada perfil. Q1, Q2 e Q3 são *range queries* e cada uma difere no tipo de computação a ser executada e no tamanho de output a ser produzido. a Q4 corresponde a uma *query* KNN simples. A *query* Q1 é a que produz um output de menor tamanho. As *queries* Q2 e Q3 produzem o mesmo volume de *output*, correspondendo a um ficheiro contendo um número de linhas igual ao número de perfis processados.

As *queries* Q1,Q2 e Q3 necessitam de um intervalo espacial / espaciotemporal como parâmetro de entrada. Devido a esse facto, e como a distribuição dos perfis Argo não é espacialmente uniforme, foram escolhidas duas áreas distintas do planeta Terra para serem utilizadas nestas *queries*, de forma a avaliar o seu desempenho em zonas com diferentes densidades de dados. A figura 4.2 contém um mapa-mundo na projeção cilíndrica de Lambert onde estão apresentadas as duas áreas escolhidas. Cada retângulo tem uma área que corresponde a 16% da área da Terra. É de realçar que as áreas escolhidas contém uma porção de área continental onde não existem perfis Argo, logo a área útil de pesquisa será menor que as das percentagens consideradas.

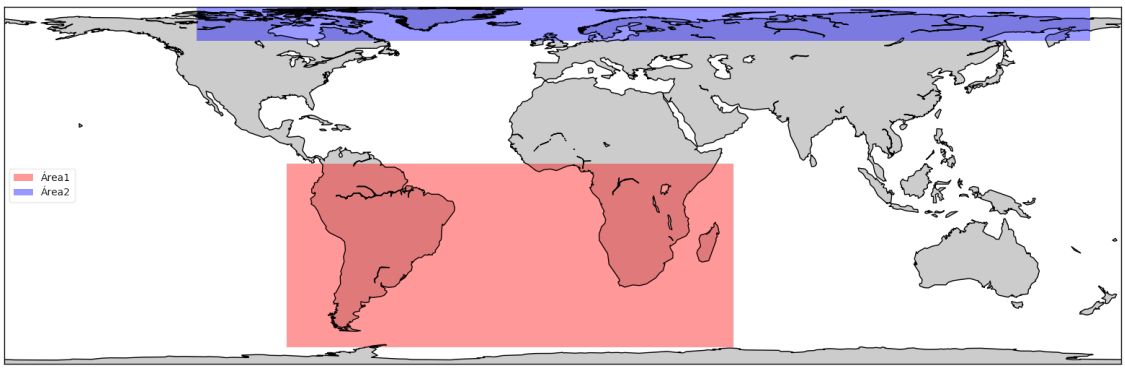


Figura 4.2: Áreas escolhidas para a parametrização de *queries*.

A área assinalada com um retângulo vermelho corresponde à área com uma maior densidade de dados, em comparação com o retângulo azul. As áreas foram ainda subdivididas em áreas mais pequenas, como se apresentam na figura 4.3, variando entre 0.003% e 16% da área total da Terra. A figura 4.4 apresenta o número de perfis contidos nas diferentes subdivisões das duas áreas escolhidas.

As pesquisas e intervalos definidos nesta secção foram utilizados para avaliar os *frameworks* Big Data escolhidos. Como o *SpatialHadoop* e *Geospark* suportam apenas indexação espacial, as pesquisas e intervalos utilizados na sua avaliação individual foram os referidos nesta secção, apenas com predicados espaciais. Na análise do *ST-Hadoop* foram

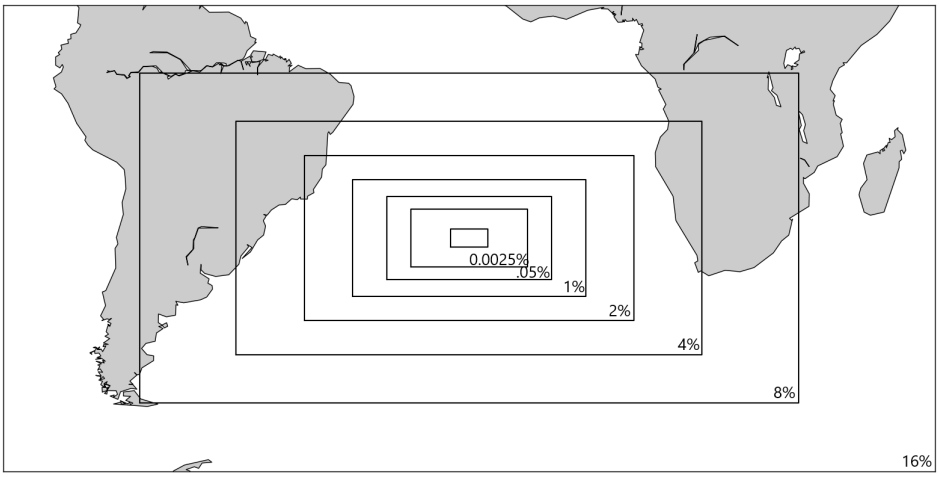


Figura 4.3: Divisão da área com maior densidade de dados.

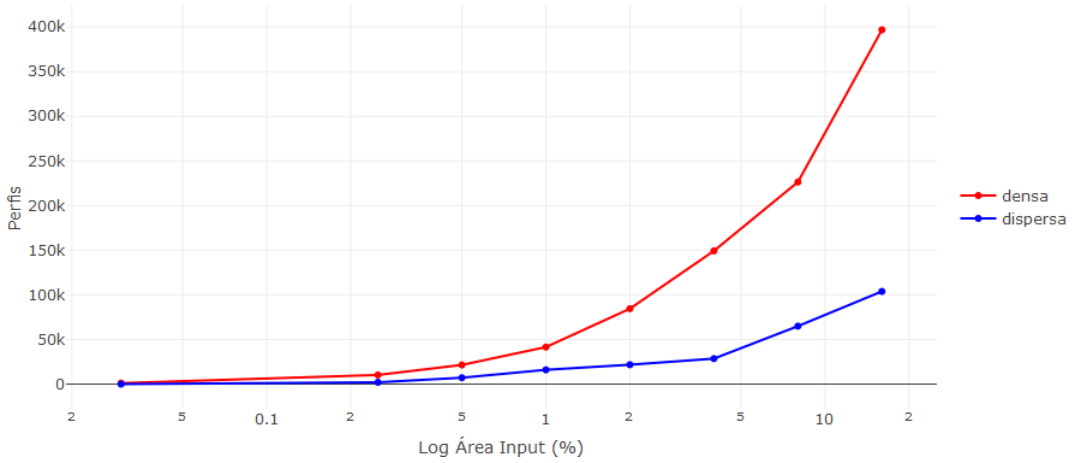


Figura 4.4: Área de queries / número de perfis.

utilizados predicados espaciais e temporais nas pesquisas, de forma a avaliar o *framework* individualmente e também em comparação com o *SpatialHadoop*.

4.4 SpatialHadoop

O primeiro passo na avaliação de criação de índices e execução de *queries* no *SpatialHadoop* consistiu no reconhecimento dos dados pelo *framework* para a criação de índices espaciais. Para tal, foi necessário criar uma classe Java que estende um dos tipos espaciais disponibilizados pelo *framework*. A classe tem como objetivo representar uma medição de um perfil de uma boia (correspondente a uma linha do ficheiro que contém os dados pré-processados) e contém ainda, métodos auxiliares que permitem a serialização dos dados a serem processados. O tipo espacial estendido foi o *Point* [2].

[2] `http://spatialhadoop.cs.umn.edu/datatypes.html`,visitado a 02/2019

Este *framework* permite apenas a criação de índices espaciais. Cada índice é constituído por um conjunto de ficheiros que correspondem a partições espaciais dos dados de entrada e um ficheiro mestre que contém informação global sobre as diferentes partições existentes.

O processo de criação de um índice espacial é constituído pelas quatro seguintes fases:

Cálculo do Minimum Bounding Rectangle: No primeiro passo é calculado o MBR dos dados de entrada providenciados. Este retângulo é calculado ao percorrer todo o input e expandir um retângulo até ser encontrado o retângulo de menor área que contenha todos os objetos geométricos percorridos.

Recolha de amostra: Após o cálculo do MBR, é colocada em memória uma amostra do input inicial (1% por defeito). A amostra é obtida ao percorrer todo o input e ao adicionar cada elemento à memória com uma probabilidade igual à percentagem de amostra pretendida.

Divisão espacial: Após a obtenção da amostra é executado o algoritmo de divisão espacial, onde são definidas as fronteiras de cada partição que irá constar no índice resultante.

Particionamento físico: Através das partições definidas no passo anterior, o input é novamente percorrido e cada elemento é colocado na partição espacial correspondente, e as diferentes partições são escritas em disco.

Associado a esta operação, também são considerados quatro parâmetros de entrada. A sua descrição e valores associados estão apresentados na tabela 4.2. A avaliação dos diferentes índices espaciais criados encontra-se na secção 4.4.1.

Após a criação e análise dos índices espaciais, as *queries* referidas na secção 4.3 foram implementadas e avaliadas sobre cada um deles. Cada *query* foi implementada de duas formas distintas:

MapReduce: Cada *query* corresponde a um programa *MapReduce* que tira partido dos índices espaciais criados.

Centralizada: A *query* é implementada com a mesma lógica que em *MapReduce*, mas o programa é apenas executado num único nó, contendo apenas paralelismo local (multithreading).

Os dois tipos diferentes de implementações procuram otimizar as operações para diferentes quantidades de dados. No processamento de uma quantidade baixa de partições de um índice, a implementação centralizada será melhor que a *MapReduce*, já que não tem *overhead* associado ao iniciar uma pesquisa. Com o aumento da quantidade de partições a processar, a versão em *MapReduce* será mais recomendável.

O racional de implementação e avaliação do desempenho das *queries* é feito na secção 4.4.2.

Tabela 4.2: Parâmetros e valores associados à criação de índices SpatialHadoop.

Propriedade	Descrição	Valores
Tamanho de bloco HDFS	Este parâmetro influencia diretamente o número de partições do índice criado, porque cada partição ocupará no máximo um bloco do HDFS em disco [22].	64, 128, 256 (MB)
Tipo de índice espacial	Este parâmetro influencia a organização das partições espaciais geradas.	Rtree(STR), Quadtree, KdTree, Z-Curve, Hilbert
% de amostra	Corresponde à percentagem de amostra utilizada para a divisão espacial dos dados. Esta percentagem manteve-se fixa nos 1%, opção baseada num estudo que concluiu que percentagens acima deste valor apresentam índices resultantes semelhantes [27].	1%
Nº de nós	Este parâmetro não está diretamente associado à operação, mas foi considerado para a avaliação da sua escalabilidade.	3, 6, 9

4.4.1 Avaliação de índices

Para a avaliação desta operação foram criados índices espaciais sobre o conjunto de dados de medições de perfis Argo, variando os parâmetros associados com os valores considerados para cada um deles (apresentados na tabela 4.2). Cada índice espacial foi apenas criado uma vez, dado o tempo de criação de cada índice ser considerável. Este *framework* suporta a inserção de dados em índices já existentes, o que minora a preocupação da recriação de índices caso existam novos dados a serem processados.

A avaliação desta operação teve em conta 5 indicadores distintos: **Tempo de criação**, **Número de Partições**, **% de Área Ocupada**, **Tamanho de partições** e **% de Sobreposição**. No restante desta secção está explicito o seu significado, bem como o impacto dos diferentes parâmetros de entrada em cada um deles. Alguns destes indicadores foram adaptados de [27], onde são analisados diferentes índices espaciais criados por este *framework*.

Na figura 4.5 é possível observar as partições dos diferentes índices espaciais onde cada linha representa um tipo de índice espacial diferente e cada coluna o tamanho de bloco do HDFS. Por baixo de cada índice é também possível observar três indicadores, onde **P** representa o número de partições, **A** a percentagem de área ocupada e **S** a sobreposição de partições.

4.4.1.1 Tempo de criação

O tempo de criação do índice contabiliza o tempo que a operação demora a ser executada. A variável que domina o tempo de criação de um índice é o número de nós do *cluster*

Figura 4.5: Visualização de índices SpatialHadoop.

onde o índice foi criado. Tal se deve ao facto de o tempo de execução desta operação ser dominado por todas as fases onde é necessário percorrer a totalidade do ficheiro de input original (recolha de Amostra e particionamento físico), que apresentam melhor desempenho com mais recursos no *cluster*.

Na figura 4.6 é possível verificar o efeito do número de nós no tempo de criação dos índices (em segundos) com diferentes algoritmos de divisão espacial (cada número de nós está representado por uma cor). Os tempos de execução demonstrados na figura não consideram o primeiro passo de criação de um índice (Cálculo do MBR), porque o MBR dos dados de entrada é fixo, sendo pré-calculado e providenciado em tempo de execução.

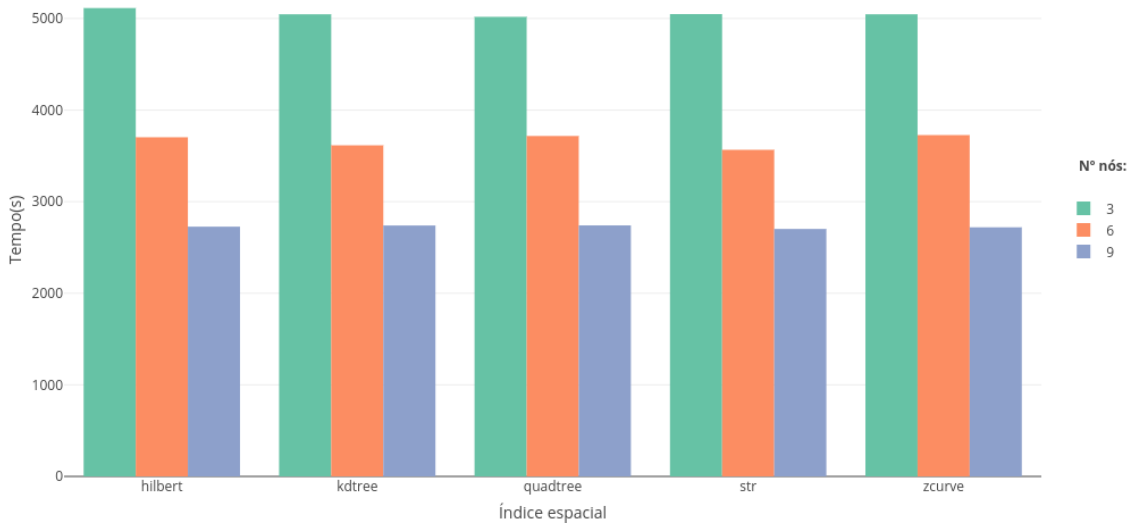


Figura 4.6: Tempo de criação de índices SpatialHadoop

4.4.1.2 Número de partições

Este indicador representa o número total de partições do índice. Os parâmetros de entrada que condicionam este indicador são o tipo de particionamento espacial escolhido e o tamanho de bloco do HDFS.

Como é possível observar nas figuras 4.5 e 4.7, quanto menor for o tamanho de bloco maior é o número de partições do índice. Também é possível observar que, de todos os tipos de índices espaciais, o índice *Quadtree* é o que cria um maior número de partições.

Este indicador é importante na realização de pesquisas, já que o número de partições a processar é um fator relevante no desempenho de uma pesquisa.

4.4.1.3 Tamanho de partições

Este indicador pretende verificar o aproveitamento do espaço em disco das partições de cada índice. Cada partição tem um limite de ocupação máxima em disco igual ao tamanho de bloco HDFS. Devido a este facto, é possível verificar para cada tipo de índice, o aproveitamento em disco das suas partições.

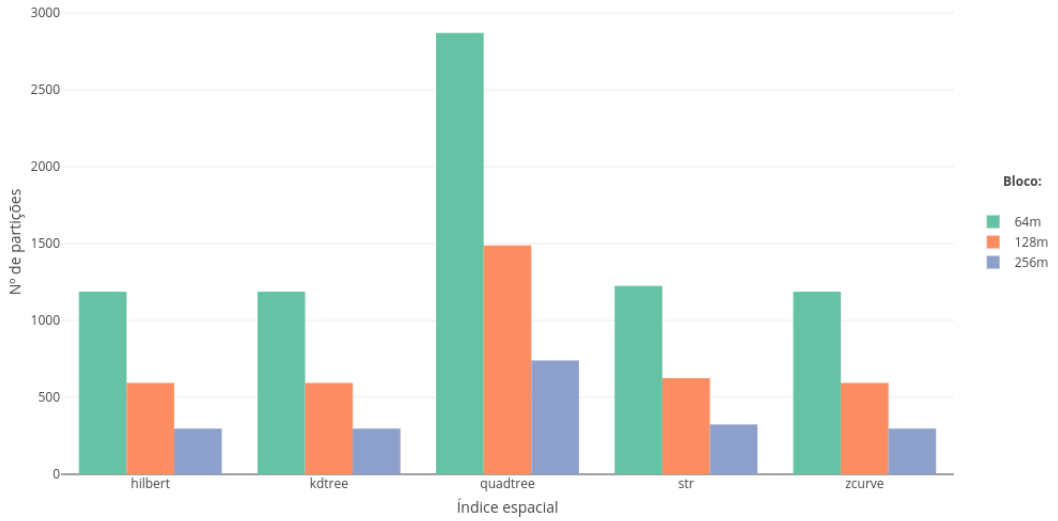


Figura 4.7: Número de partições de índices SpatialHadoop

O gráfico 4.8 apresenta a distribuição dos tamanhos das partições de cada índice espacial com os diferentes tamanhos de bloco HDFS. No geral, as partições dos diferentes índices apresentam um espaço em disco próximo do limite associado. Apenas as partições do índice *quadtree* apresentam uma utilização de disco bastante inferior às restantes.

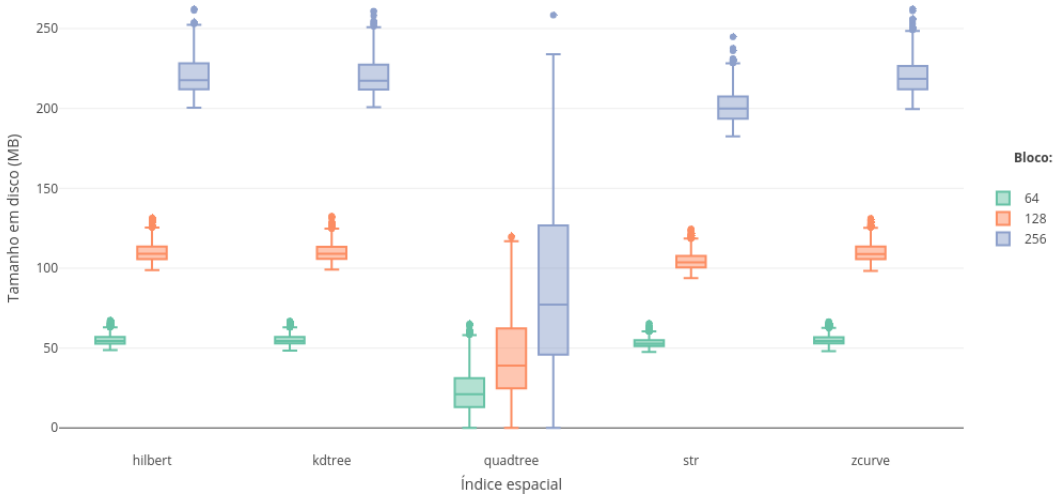


Figura 4.8: Espaço ocupado em disco de índices SpatialHadoop

4.4.1.4 % de Área ocupada

Este indicador é calculado ao somar o total de área ocupada pelas partições do índice e dividi-la pela área do MBR do conjunto de dados original. É pretendido indicar o ajuste espacial das partições do índice à distribuição dos dados que este contém.

Este indicador poderá ser importante otimizar já que um maior ajuste espacial diminui o tempo de resposta a pesquisas que contenham intervalos espaciais que não se

sobreponham com nenhuma partição existente.

Nas figuras 4.5 e 4.9 é de notar que os índices *zcurve e Hilbert* apresentam valores acima dos 100% devido à sobreposição entre partições. O índice *Quadtree* embora tenha um maior número de partições e ocupação de disco por partição menor que os restantes algoritmos, é o que contem um melhor aproveitamento espacial.

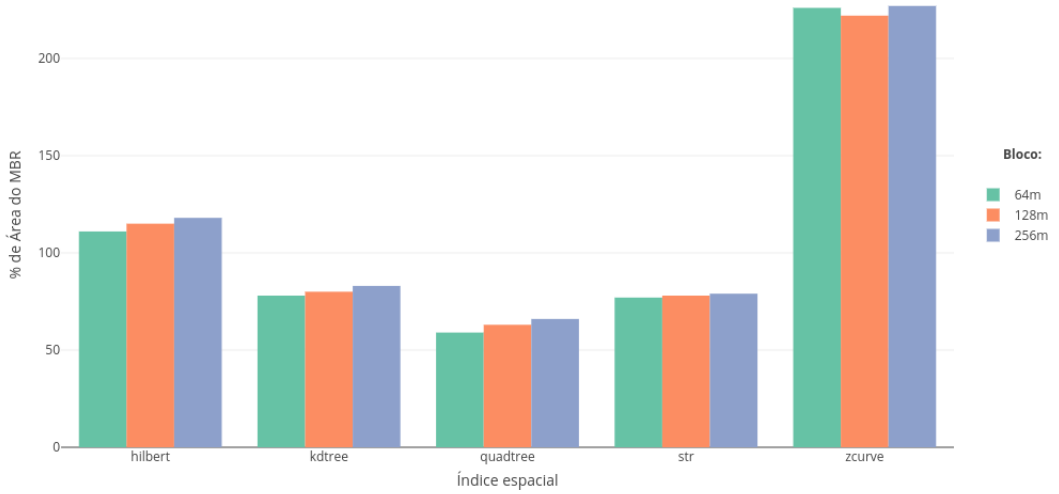


Figura 4.9: % da área do MBR ocupada pelas partições de índices SpatialHadoop

4.4.1.5 % de Área de sobreposição

Este indicador foi utilizado para verificar a existência de sobreposição entre partições de um índice. Foi calculado ao somar o total da área de sobreposição entre partições, e depois dividir pela área do MBR do conjunto de dados. Como é possível observar na figura 4.5, apenas os índices (Hilbert e Z-Curve) possuem sobreposição entre partições.

4.4.2 Avaliação de queries

Na fase de avaliação das *queries*, estas foram em primeiro lugar implementadas através dos mecanismos providenciados pelo *SpatialHadoop*. A implementação das *queries* Q1, Q2 e Q3 teve como base a *range query* providenciada pelo SpatialHadoop, que filtra as partições do índice espacial ao verificar as que se sobrepõem com o intervalo espacial fornecido. A implementação da *query* Q4 foi a disponibilizada pelo *framework*. Consiste num algoritmo iterativo que a cada iteração procura os K elementos mais próximos de um ponto X (definido pelo utilizador), que se encontrem em partições que estejam num raio Y desse mesmo ponto. Caso não sejam encontrados K elementos nas partições processadas, é iniciada uma nova iteração com $Y = Y \times 2$. Os valores de Y indicados são os considerados por defeito, podendo ser alterados pelo utilizador.

Para cada *query* de avaliação implementada foi medido o tempo médio de 10 execuções sobre cada índice espacial criado.

No restante desta secção são apresentadas as conclusões referentes a três fatores que demonstraram maior impacto na execução de *queries*:

Tamanho de bloco: Com a análise a esta variável, é procurado verificar o impacto do balanço entre o número de partições e o tamanho de cada partição de um índice espacial no processamento de uma *query*.

Centralizado vs MapReduce: Como referido na secção 4.4, as *queries* de avaliação foram implementadas de forma centralizada (executadas num único nó) e também de forma a utilizar *MapReduce*. Foi procurado avaliar o impacto do tipo de implementação no desempenho das diferentes *queries*.

Nº de nós: Foi avaliada a escalabilidade horizontal das operações ao adicionar nós ao *cluster* utilizado.

4.4.2.1 Tamanho de bloco / Nº de Partições

Nas *queries* **Q1, Q2 e Q3**, os índices com menor tamanho de bloco apresentam melhor desempenho para áreas de pesquisa muito pequenas. Em áreas maiores, índices com maior tamanho de bloco (menor número de partições) apresentam melhor desempenho. Nas figuras 4.10 e 4.11 são apresentados os tempos de execução da *query* **Q2** para o índice *R-tree(str)* nas duas diferentes áreas consideradas, onde é possível observar o comportamento referido.

Como o output produzido nas *queries* Q2 e Q3 aumenta consoante a área de pesquisa, o tempo de execução é afetado pelo tempo de produção dos resultados. Para avaliar o impacto do output, foi efetuada a divisão entre o número de perfis presentes no ficheiro de output e o tempo de execução de cada pesquisa. As figuras 4.12 e 4.13 demonstram o rácio para a *query* Q2, onde é possível observar o mesmo comportamento referido no parágrafo anterior.

No caso da *query* **Q4** verifica-se o inverso. Os índices com menor tamanho de bloco apresentam melhor desempenho, com a exceção do índice *Quadtree* que tem um elevado número de partições, obrigando a serem executadas várias iterações desta pesquisa para serem encontrados os K elementos. Na figura 4.14 estão apresentados os tempos de execução da *query* Q4 para o índice R-tree(str), na área com maior densidade de dados.

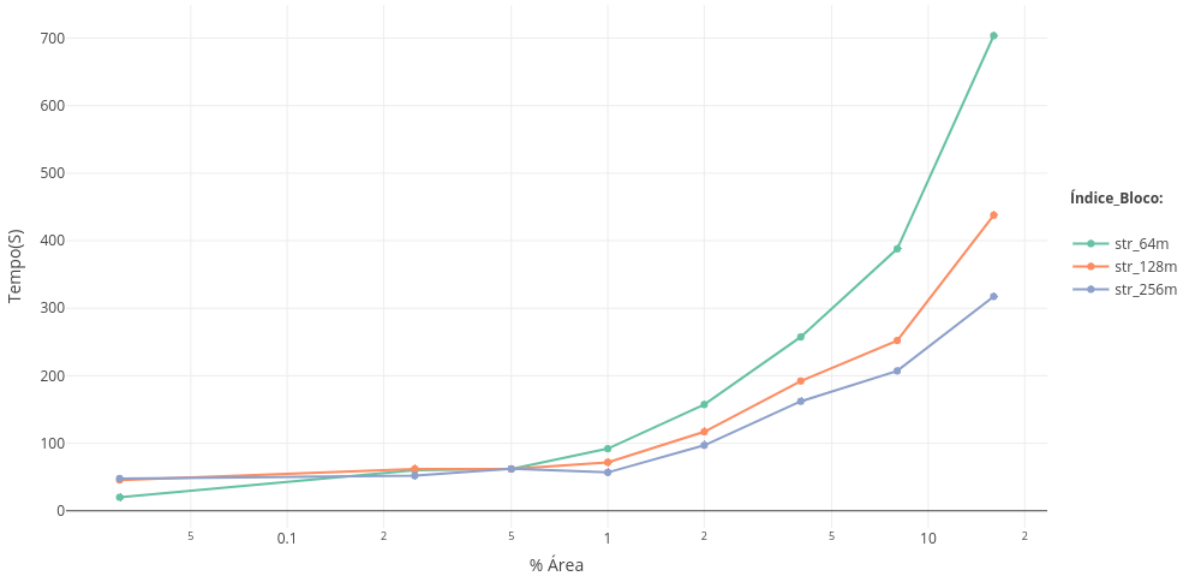


Figura 4.10: Execução de Q2(MapReduce) em índices *SpatialHadoop* num *cluster* de 3 nós (Índice R-tree, Variação de tamanho de bloco, área densa)

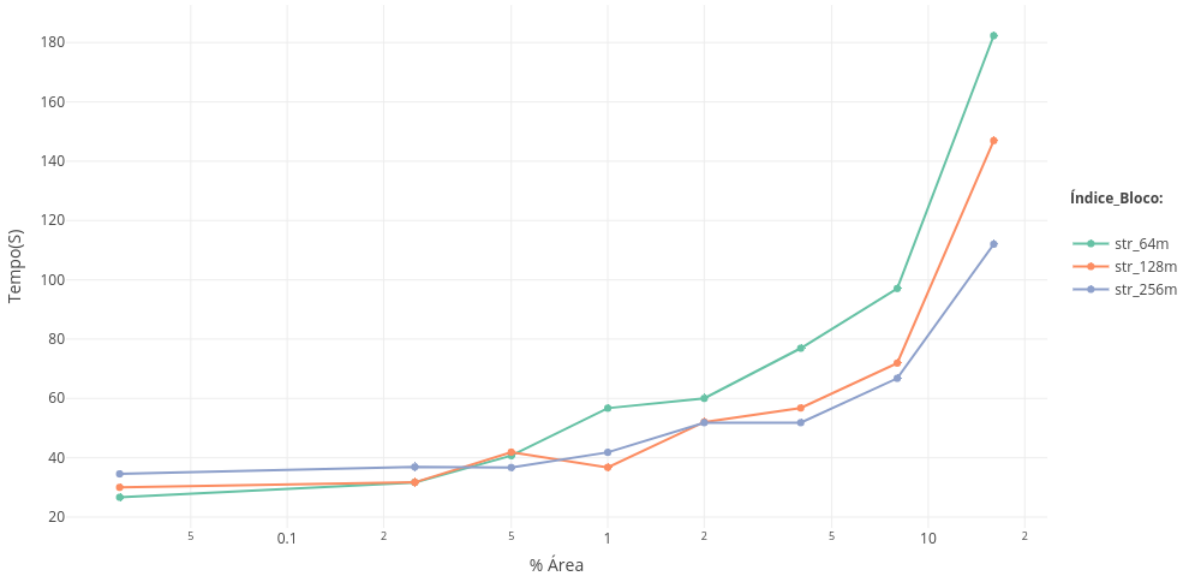


Figura 4.11: Execução de Q2(MapReduce) em índices *SpatialHadoop* num *cluster* de 3 nós (Índice R-tree, Variação de tamanho de bloco, área dispersa)

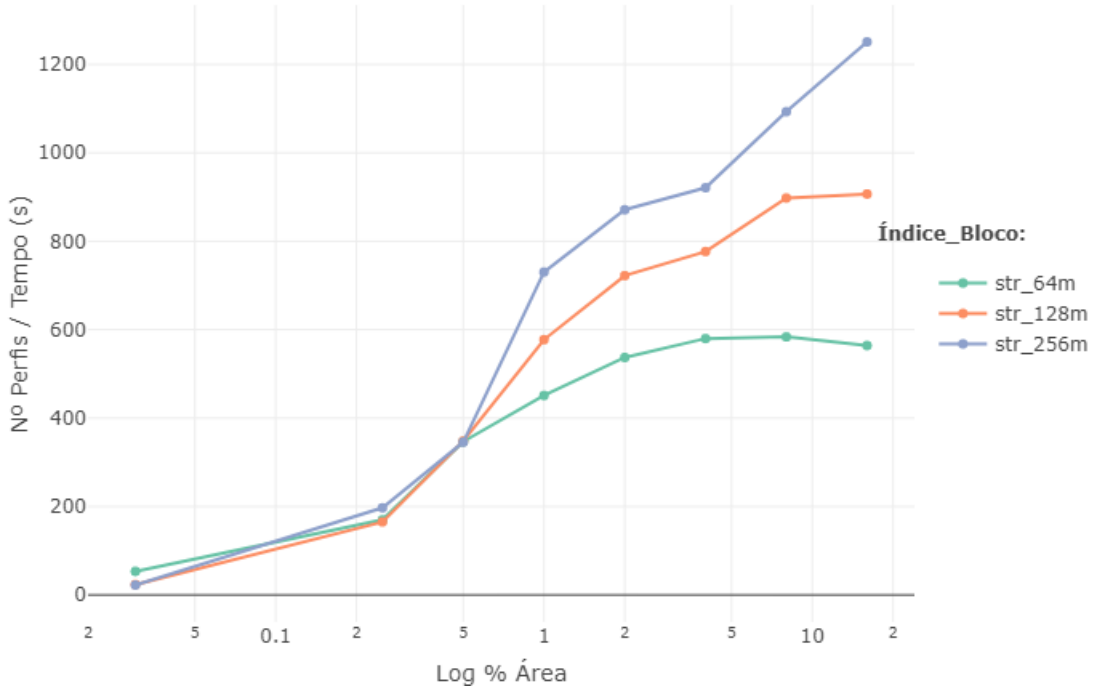


Figura 4.12: Rácio entre nº elementos de output e tempo de execução de Q2 de Q2(MapReduce) em índices *SpatialHadoop* num *cluster* de 3 nós (Índice R-tree, Variação de tamanho de bloco, área densa)

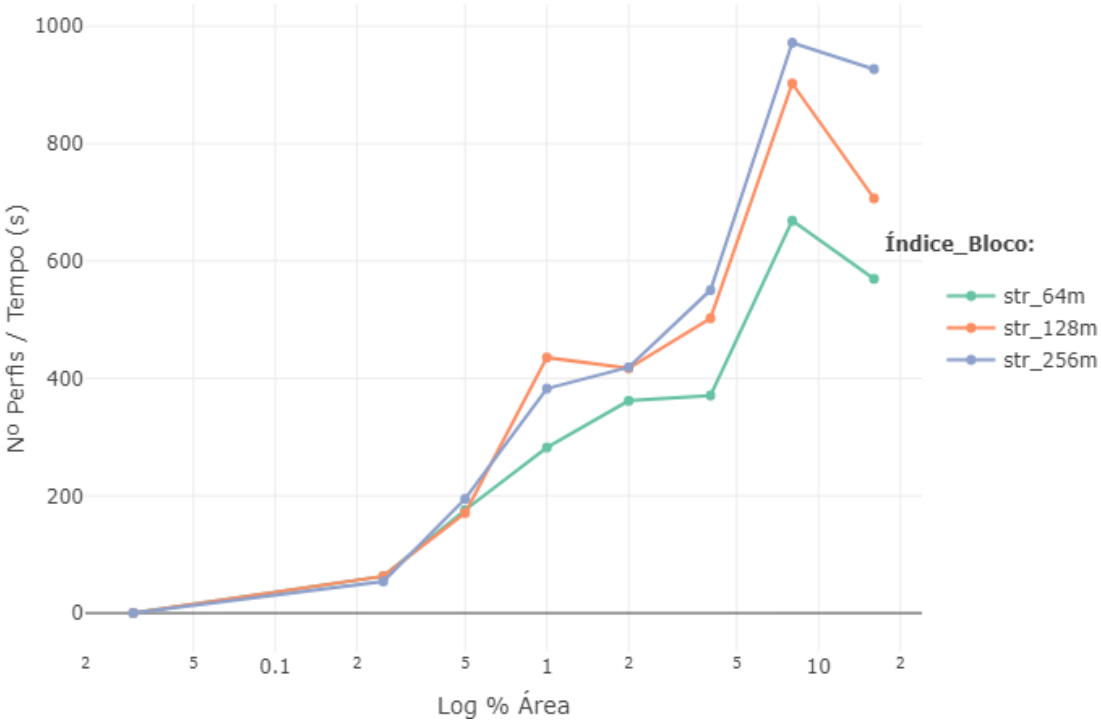


Figura 4.13: Rácio entre nº elementos de output e tempo de execução de Q2(MapReduce) em índices *SpatialHadoop* num *cluster* de 3 nós (Índice R-tree, Variação de tamanho de bloco, área dispersa)

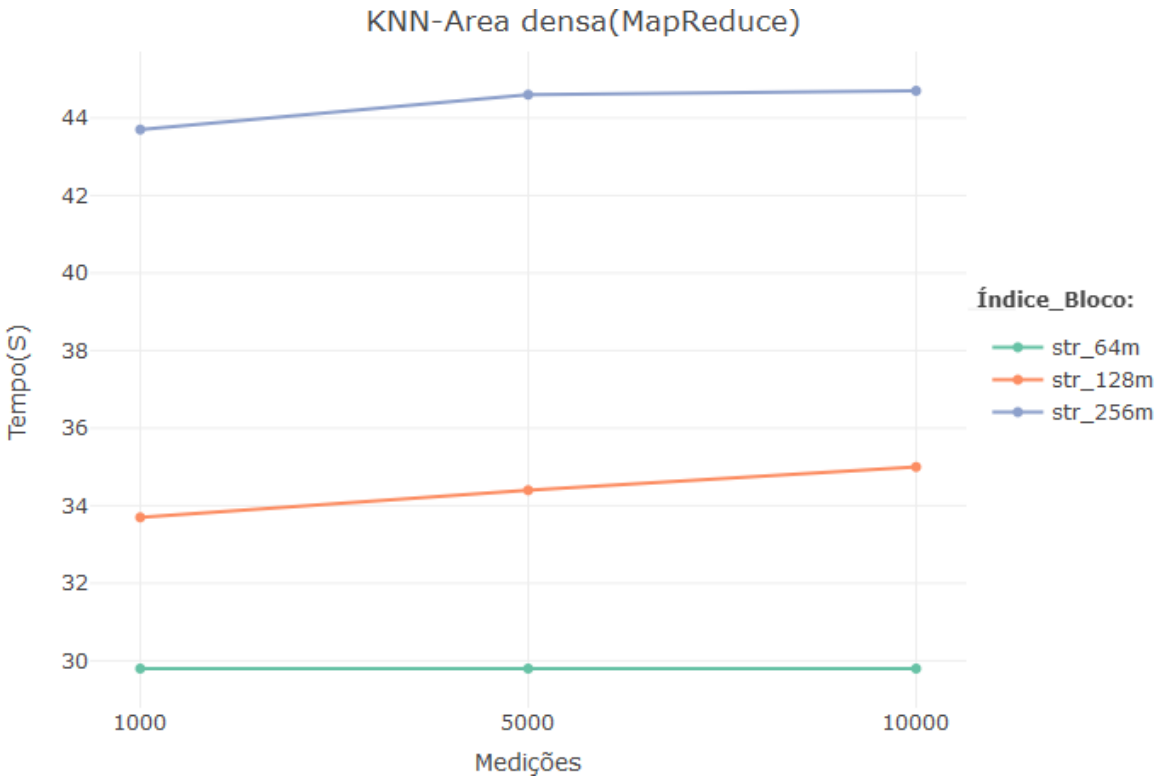


Figura 4.14: Execução de Q4(MapReduce) em índices *SpatialHadoop* num *cluster* de 3 nós (Variação de tamanho de bloco, área densa)

4.4.2.2 Centralizado vs MapReduce

Nas *queries* **Q1, Q2, Q3** em áreas mais pequenas (menor número de partições de um índice a processar) a versão centralizada tem um melhor desempenho. Com o aumentar da área (mais partições a processar) a versão *MapReduce* acaba por apresentar melhor desempenho na *query* mais exigente a nível de processamento das medições. Esta observação foi verificada em todos os índices espaciais estudados.

A figura 4.15 demonstra o caso onde tal ocorreu, na execução da *query* Q3 sobre um índice *R-tree* (bloco 128m) nas duas versões (MapReduce em cluster com 3 nós), onde é possível observar que o tempo de execução nos 16% de área apresenta melhor desempenho que a versão centralizada.

Na *query* Q4, devido à sua natureza iterativa, a versão centralizada apresentou sempre um desempenho bastante superior à versão *MapReduce*, que tem uma sobrecarga associada ao iniciar uma nova iteração. Este efeito foi maioritariamente visível em pesquisas na área com menor densidade de dados, onde foram necessárias várias iterações para encontrar o resultado final.

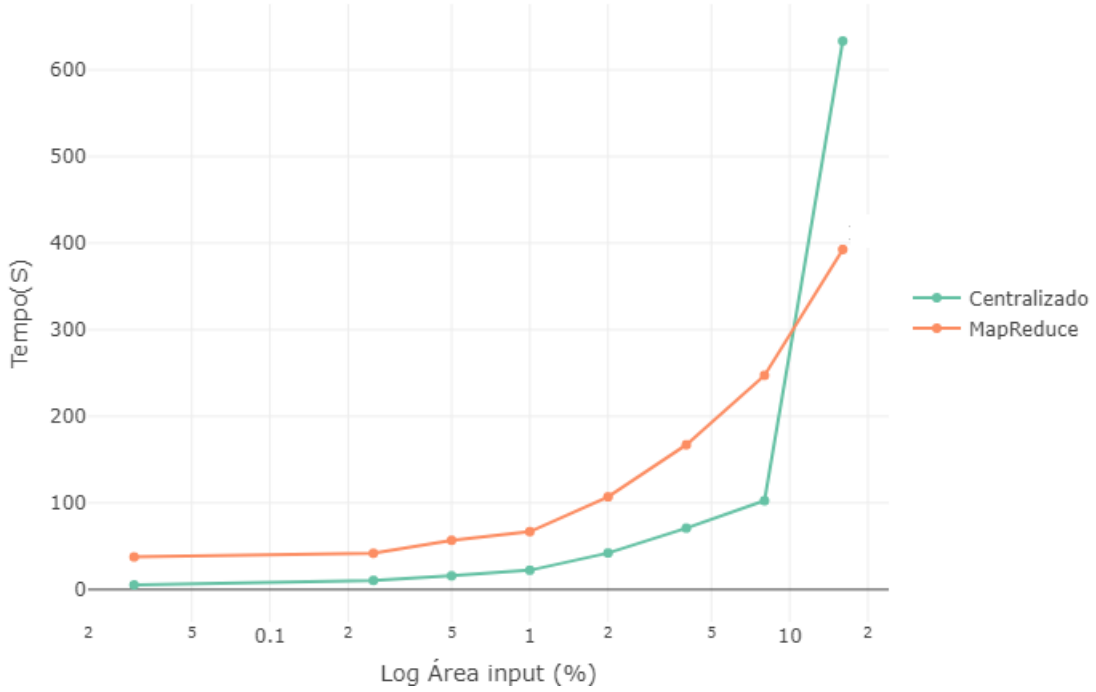


Figura 4.15: Execução de Q3(MapReduce vs Centralizado) em índice *R-tree SpatialHadoop* num *cluster* de 3 nós (área densa)

4.4.2.3 Nº de nós do cluster

Para verificar a escalabilidade das *queries* de avaliação, cada uma foi executada em *clusters* com diferentes números de nós (3, 6, 9) disponíveis. Para áreas pequenas, o efeito de escalabilidade não é visível porque os recursos do *cluster* não são totalmente utilizados. A

partir dos 8% de área considerada, já é possível observar o efeito do aumento do número de nós no desempenho das *queries* avaliadas. Este efeito foi verificado em todos os índices espaciais considerados.

Nas figuras 4.17, 4.16 e 4.18 estão apresentados os tempos de execução das *queries* Q1, Q2 e Q3, que foram executadas sobre o índice R-tree com diferentes tamanhos de bloco, e numa área de 16% na zona com maior densidade de dados.

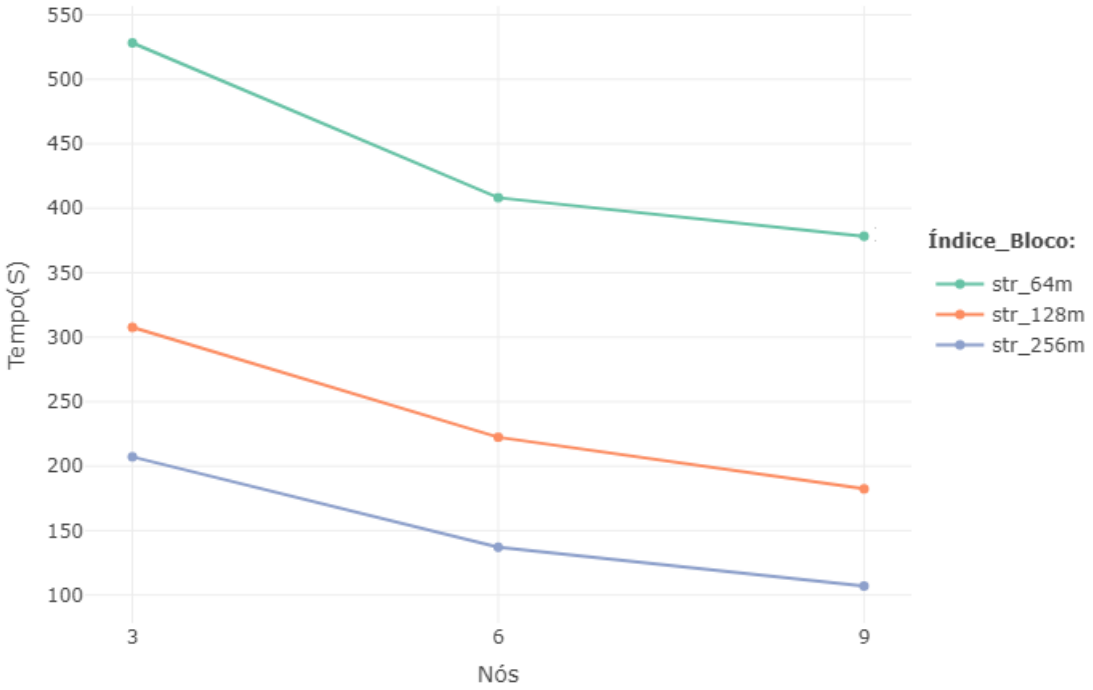


Figura 4.16: Verificação de escalabilidade de Q1(MapReduce) em índices *RTree* num *cluster* de 3 nós (área densa, 16%)

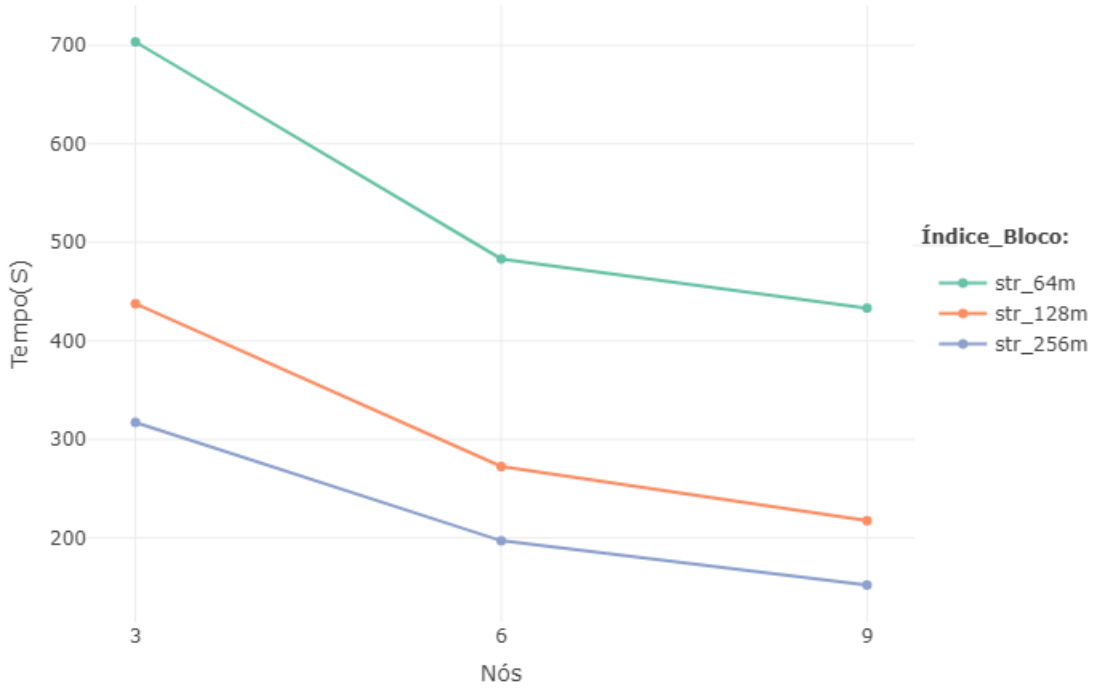


Figura 4.17: Verificação de escalabilidade de Q2(MapReduce) em índices *RTree* num *cluster* de 3, 6 e 9 nós (área densa, 16%)

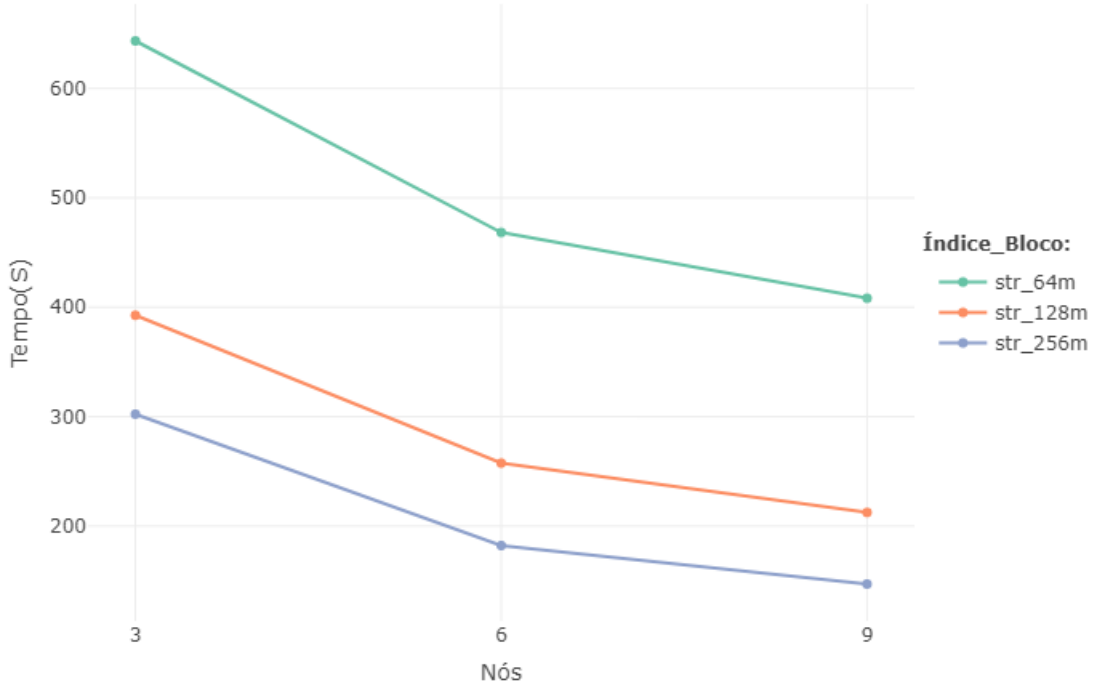


Figura 4.18: Verificação de escalabilidade de Q3(MapReduce) em índices *RTree* num *cluster* de 3, 6 e 9 nós (área densa, 16%)

4.5 ST-Hadoop

Apresenta-se agora a avaliação do *ST-Hadoop*. Este *framework* estende o *SpatialHadoop* para suportar o processamento de dados espaciotemporais.

O reconhecimento dos dados por parte do *framework* para a criação de índices, é semelhante ao referido na secção 4.4 mas foi estendido o tipo de dados *STPoint* [3], que contém informação sobre duas coordenadas de localização e um *timestamp* associado.

O índice espaciotemporal é composto, em primeiro lugar, por uma divisão por granularidade temporal. Dentro de cada granularidade estão os diferentes níveis associados e em cada nível encontra-se um índice espacial com a estrutura indicada em 4.4. O processo de criação de um índice numa determinada granularidade temporal, consiste nos seguintes passos:

Divisão Temporal O input fornecido é dividido em várias partes, cada parte contendo dados correspondentes a um nível da granularidade temporal providenciada pelo programador.

Criação de Índices Espaciais Após o input ser dividido nas diferentes partes, é construido um índice espacial utilizando cada uma delas, seguindo os passos referidos em 4.4.

Este processo pode ser repetido para diferentes granularidades temporais, introduzindo replicação dos dados mas com uma melhor capacidade de resposta a diferentes *queries* temporais.

Os parâmetros de entrada considerados nesta operação, são os referidos na secção 4.4 e ainda a escolha de granularidade temporal que pode conter os seguintes valores: **(segundo, minuto, hora, dia, mês, ano)**. A avaliação dos índices criados encontra-se na secção 4.5.1.

Após a criação dos índices espaciotemporais, o objetivo na avaliação de *queries* residiu em dois pontos:

- Verificar o impacto da camada temporal dos índices ST-Hadoop no desempenho de *queries* espaciais em comparação com índices SpatialHadoop.
- Verificar desempenho ao executar *queries* com diferentes predicados temporais em índices *ST-Hadoop* e *SpatialHadoop*,

A análise aos pontos referidos acima encontra-se na secção 4.5.2.

4.5.1 Avaliação de Índices

Devido à larga extensão temporal dos dados a serem processados (entre os anos 1997 e 2018), foi apenas criado um índice espaciotemporal com o grau de granularidade temporal **ano**. Isto deve-se ao elevado tempo de execução operação, que mostrou ser 5 a 6 vezes

[3] http://st-hadoop.cs.umn.edu/getting-started/extendable-st-shape, visitado a 02/2019

superior aos índices *SpatialHadoop*. A variação dos parâmetros escolhida para a criação do índice está apresentada na tabela 4.3.

Tabela 4.3: Parâmetros e valores associados à criação de índices ST-Hadoop.

Parâmetro	Valor
Tamanho de bloco HDFS	128(MB)
Índice Espacial	Rtree(STR)
Nº de nós	6
Amostra	1(%)
Granularidade Temporal	Ano

O tempo de criação deste índice foi de **5 horas e 35 minutos** num *cluster* com 6 nós.

4.5.2 Avaliação de Queries

4.5.2.1 Pesquisas espaciais

Para avaliar a sobrecarga que a camada temporal do índice ST-Hadoop induz em pesquisas com apenas predicados espaciais (foi considerada toda a extensão temporal dos dados), foram executadas as *queries* **Q1,Q2 e Q3** (Versão *MapReduce*) sobre o índice espaciotemporal criado e um índice SpatialHadoop com parâmetros equivalentes. Cada *query* considerada foi executada 10 vezes num *cluster* com 6 nós disponíveis.

Como é possível observar nas figuras 4.19 e 4.20, o índice espaciotemporal apresenta pior desempenho. Tal ocorre devido à forma de como as *queries* são executadas em índices *ST-Hadoop*. No *ST-Hadoop* quando um predicado temporal envolve vários níveis, é executada uma pesquisa para cada um deles, sendo necessário aceder a vários índices espaciais localizados a cada nível considerado dentro de uma granularidade temporal. No caso desta avaliação, isto significa que para responder às *queries* espaciais foi necessário executar 21 pesquisas, cada uma referente a um ano da extensão temporal dos dados (1997-2018). Assim é possível concluir que o desempenho de uma determinada *query* espacial/espaciotemporal é diretamente impactado pela capacidade de resposta a *queries* simultâneas de um *cluster*.

A *query* Q4 não foi avaliada porque não é disponibilizada com parametrizações espaciotemporais pelo ST-Hadoop.

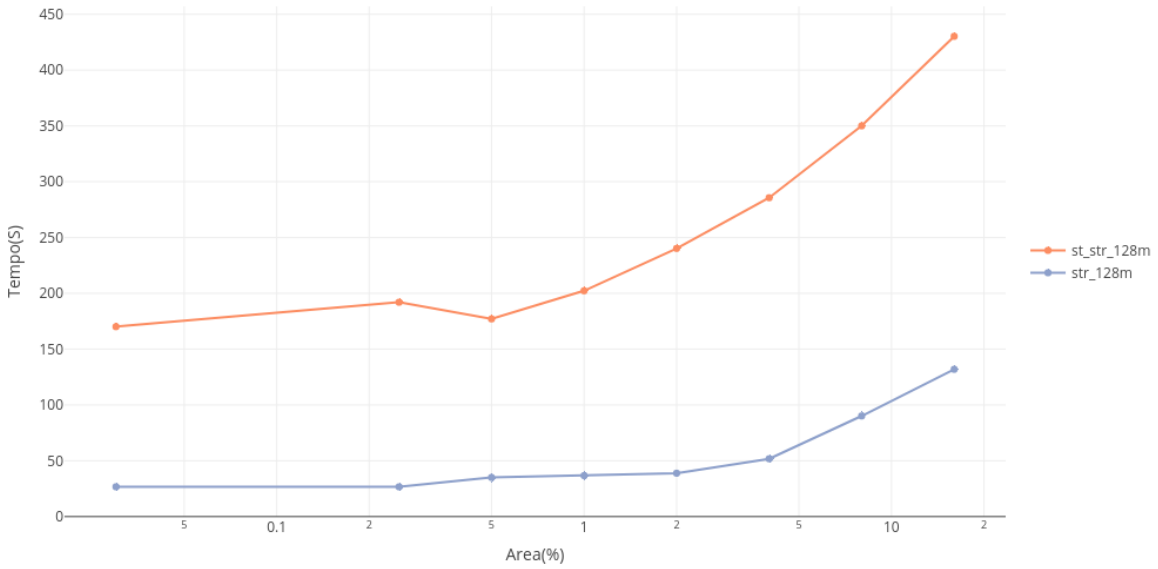


Figura 4.19: Comparação de pesquisas espaciais (Q1) em índices St-Hadoop (laranja) e SpatialHadoop (azul) em área dispersa

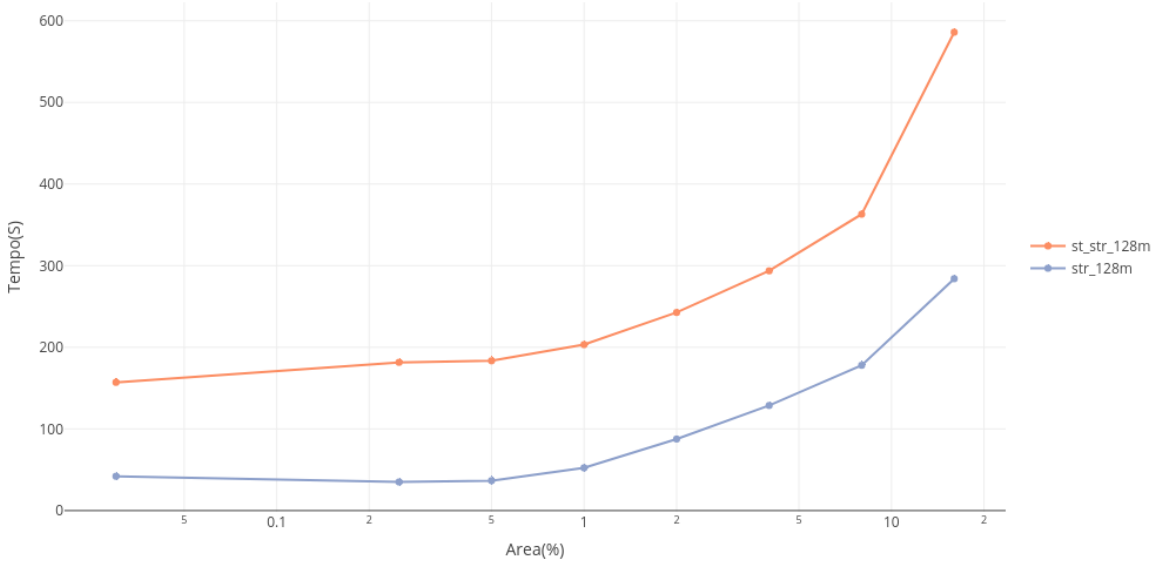


Figura 4.20: Comparação de pesquisas espaciais (Q1) em índices St-Hadoop (laranja) e SpatialHadoop(azul) em área densa

4.5.2.2 Pesquisas espaciotemporais

Para avaliar o ganho de desempenho que a camada temporal do índice espaciotemporal providencia, foram adicionados diferentes intervalos temporais às *queries* Q1, Q2 e Q3. Na figura 4.21 é apresentado o desempenho da pesquisa Q1 com diferentes intervalos temporais em índices SpatialHadoop. Foram considerados quatro intervalos anuais: 2018, 2017-2018, 2016-2018, 2015-2018. O desempenho é semelhante para todos os intervalos,

porque, como existe apenas divisão espacial, para obter o resultado da *query* é necessário aceder à mesma quantidade de dados independentemente do predicado temporal fornecido.

A figura 4.22 demonstra o mesmo que a anterior mas no índice ST-Hadoop. Como é possível observar, intervalos menores apresentam melhor desempenho porque apenas são acedidos os dados contidos no intervalo temporal correspondente. É também importante realçar que a capacidade inicial de resposta a cada *query* está diretamente relacionada com a capacidade de resposta *cluster* a *queries simultâneas*. Como é possível observar no aumento do tempo de resposta nas menores percentagens de área consideradas, com o aumento de anos a processar.

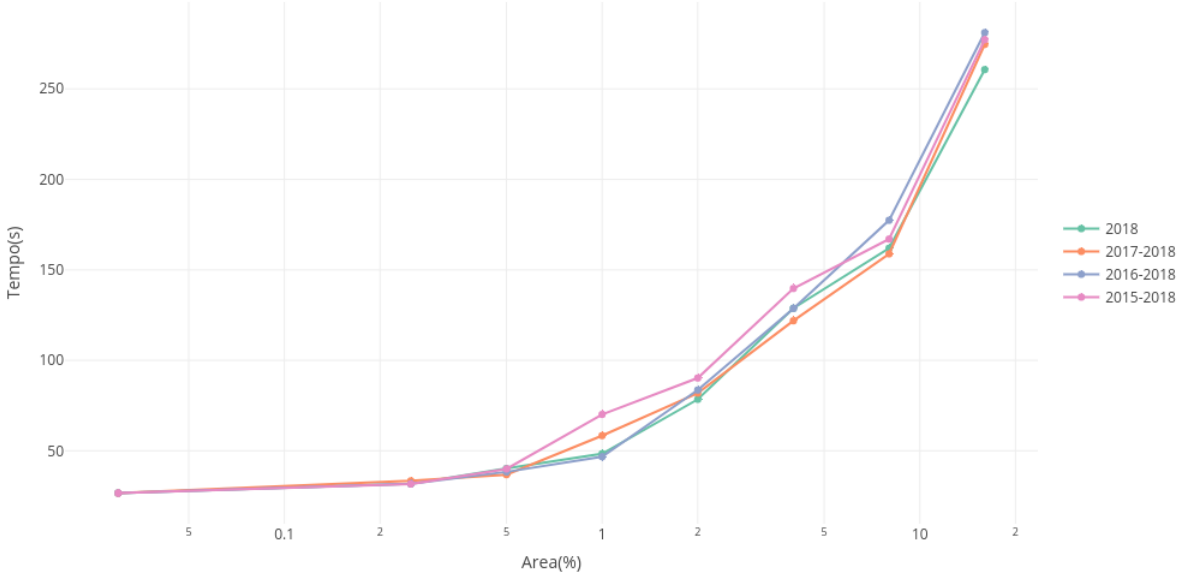


Figura 4.21: Comparação de pesquisas espaciotemporais (Q1) em índices Rtree SpatialHadoop, em área densa

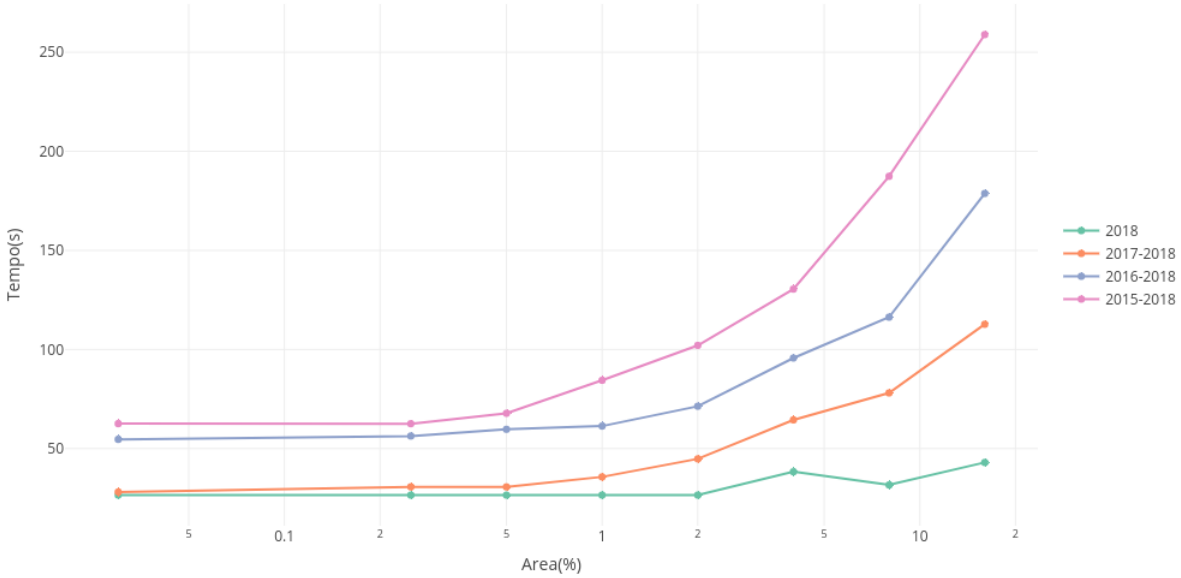


Figura 4.22: Comparação de pesquisas espaciais (Q1) em índices Rtree St-Hadoop(laranja), em área densa

4.6 Geospark

Por fim, apresentam-se os resultados para o *Geospark*. Este difere das restantes porque é baseada em *Apache Spark* ao invés de *MapReduce*.

O primeiro passo consistiu no carregamento do conjunto de dados para um *Spatial Resillient Distributed Dataset*, onde foram tidos dois aspetos importantes em conta: O número de partições e a forma de persistência do SRDD.

O número mínimo de partições consideradas para um RDD corresponde ao número de blocos que o ficheiro de input contém no HDFS que, no caso dos dados de input utilizados, corresponde a 475 partições. Nos testes realizados, foi constatado que vários executores falhavam ao tentar carregar o SRDD. Foram então seguidas recomendações de aumentar o número de partições 2 a 8 vezes mais do valor original [4], e a partir de 1000 partições já não se verificaram falhas de executores no carregamento dos dados.

O outro aspeto considerado, consistiu na escolha da forma de persistência do SRDD a ser carregado. Cada escolha representa uma troca entre o uso de memória e eficiência a nível de *CPU* de um *cluster*, no armazenamento e execução de operações em *RDD*s. Os níveis de persistência considerados foram os seguintes:

MEMORY_ONLY: O RDD é guardado como um conjunto de objetos Java não serializados em memória. Se o RDD não couber em memória, algumas partições serão recalculadas sempre que necessário.

[4] http://datasystemslab.github.io/GeoSpark/tutorial/Advanced-Tutorial-Tune-your-GeoSpark-Application/,visita 02/2019

MEMORY_AND_DISK: O RDD é guardado como um conjunto de objetos Java não serializados em memória. Se o RDD não couber em memória, algumas partições serão guardadas em disco.

MEMORY_ONLY_SER: O RDD é guardado como um conjunto de objetos Java serializados (1 byte array por partição) em memória. Esta opção fará o *RDD* ocupar menos espaço que a opção *MEMORY_ONLY*, mas necessitará de mais recursos *CPU* para a sua leitura.

Tendo em conta as características do *cluster* disponível (secção I.2.2) a terceira opção apresentou ser a melhor, porque permitiu a persistência do *SRDD* totalmente em memória. Na tabela 4.4 está apresentado o espaço ocupado em memória e disco do RDD com as diferentes opções de persistência.

Tabela 4.4: Espaço ocupado em memória e disco por um SRDD

Tipo	Input	Memória	Disco	Recálculo de Partições
MEMORY_ONLY	61GB	63.5GB	0 GB	Sim
MEMORY_AND_DISK	61GB	63.5GB	46.7 GB	Não
MEMORY_ONLY_SER	61GB	63.5GB	0 GB	Não

O processo de criação de índices neste*framework* consiste em criar um índice espacial local em cada partição do SRDD. A sobrecarga de armazenamento associada ao persistir estes índices, não permitiu colocar a totalidade do conunto de dados em memória. Logo, não foi possível verificar o possível melhoramento de desempenho das diferentes *queries* de avaliação. Foi constatado noutra análise a este *framework* [26], que a utilização destes índices apresenta melhorias de desempenho em operações do tipo *Join*, que não foram estudadas nesta avaliação.

Por fim, foram utilizadas as implementações de *range e KNN queries* disponíveis nos SRDD para a implementação das *queries* de avaliação.

4.6.1 Avaliação de Queries

As diferentes *queries* foram executadas sobre o SRDD persistido de forma *MEMORY_ONLY_SER*, onde foram medidos os tempos médios de 10 execuções. Na tabela 4.5 são apresentados os parâmetros *Spark* associados, seguindo as recomendações discutidas na secção I.2.2.

Tabela 4.5: Propriedades *Spark* utilizadas na execução de *queries*

Propriedade	Valor
num-executors	11
executor-memory	12 GB
executor-cores	5
RDD partitions	1000, 1500, 2000

No geral, o número de diferentes partições não afetou o desempenho das pesquisas. Nas *queries* Q1, Q2, Q3 o tempo de execução foi fixo em qualquer área considerada. O tempo de execução médio foi **12 segundos** num SRDD com 1000 partições, sendo que, os outros valores de partições consideradas apresentam tempos de execução idênticos.

Na *query* Q4, foi possível verificar que o aumento do parâmetro K impacta o seu desempenho. Tal ocorre porque os K vizinhos são encontrados, ao primeiro juntar os K elementos de cada partição existente, ordená-los por distância, e por fim devolver os K elementos mais próximos. O tempo de ordenação estará relacionado com o valor K escolhido. Na figura 4.23 estão apresentados os tempos de execução de Q4 num SRDD com 1000 partições, para diferentes valores de K.

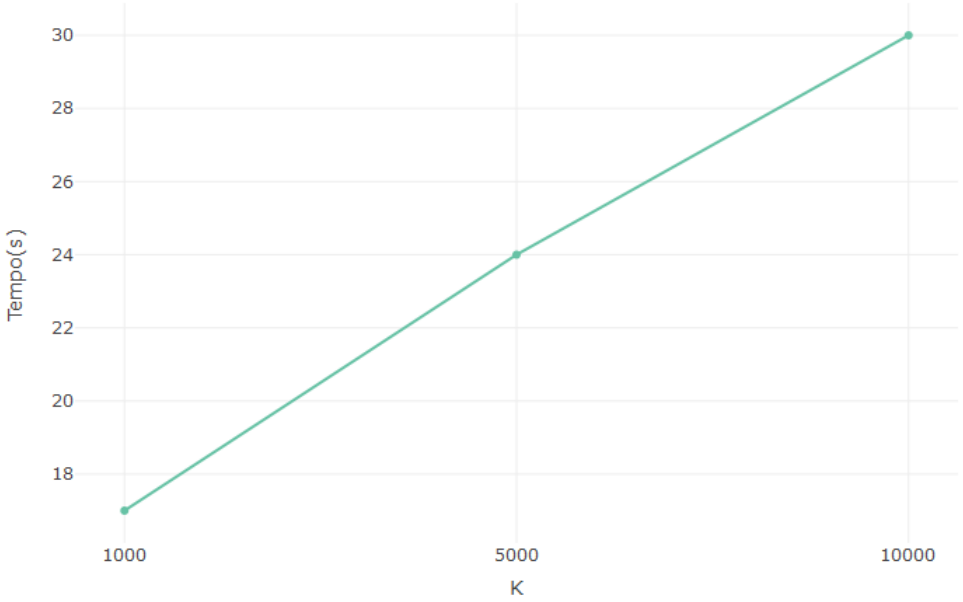


Figura 4.23: Tempos de execução de Q4 sobre o *SRDD* com 1000 partições

4.7 Conclusões

Neste capítulo foram avaliados experimentalmente três *frameworks* com o objetivo de armazenar e processar o conjunto de dados de medições Argo.

No *framework* SpatialHadoop, foi verificada a importância da parametrização na criação de índices e como estes afetam o desempenho das diferentes *queries* espaciais. A escolha de tamanho de bloco e tipo de divisão espacial controlam diretamente o número e organização das partições de um índice. Foi também verificado que em pesquisas que envolvam poucas partições de um índice, uma versão centralizada é preferível a *MapReduce*. Por fim, também foi avaliada a escalabilidade horizontal destas operações e como o seu desempenho é afetado pelo aumento de recursos de uma infraestrutura.

Na análise do *framework* ST-Hadoop foi constatado que a divisão temporal é um aspeto importante no tempo de criação de índices. Na avaliação de *queries* foi analisada

a melhoria de desempenho em relação a índices SpatialHadoop na execução de *queries* espaciotemporais. Foi possível demonstrar que para pesquisas espaciais, a divisão temporal apresenta um *overhead* considerável. Em pesquisas espaciotemporais, os índices ST-Hadoop apresentam melhor desempenho dependendo da capacidade do *cluster* responder paralelamente a *queries* e da extensão temporal a ser considerada.

Finalmente, na análise do *framework* Geospark, foi possível constatar a importância de carregar o SRDD da forma mais eficiente tendo em conta a infraestrutura disponível de forma a otimizar o desempenho das diferentes operações.

Como resultado desta avaliação, foi evidente que o *framework Geospark* apresenta uma vantagem considerável no tempo de processamento de diferentes pesquisas sobre os dados Argo, caso seja possível o seu processamento total em memória. Embora os *frameworks SpatialHadoop* e *ST-Hadoop* não contenham essa possibilidade (visto utilizarem *MapReduce* como base de processamento), apresentam alguma flexibilidade ao permitirem a execução de *queries* localmente. A escolha de um índice apropriado e um balanço adequado entre a utilização de *Mapreduce* ou processamento local, tendo em conta a infraestrutura disponível e os dados a processar, é fundamental para obter o melhor desempenho possível ao utilizar estes *frameworks*.

Capítulo 5

Implementação de caso de uso

Nesta secção é descrita a implementação de um caso de uso real sobre perfis Argo através do uso dos *frameworks* SpatialHadoop e ST-Hadoop.

O caso de uso escolhido foi a criação do produto *ArgoMixedLayers* (descrito na secção 2.3.3.2). Este caso de uso foi implementado devido à disponibilização de um manual de criação bem como de código fonte, tornando viável a sua implementação correta.

Este produto disponibiliza dois componentes distintos:

- Base de dados que contém a profundidade da camada mista de cada perfil Argo existente, calculado através de dois algoritmos [12, 13].
- Climatologia que agrega os valores dos diferentes perfis em divisões espaciotemporais de $1° \times 1°$ de latitude e longitude, para cada mês do ano.

A primeira parte desta implementação focou-se no desenvolvimento de um dos algoritmos para o cálculo da camada mista num determinado perfil utilizando a linguagem Java. A implementação deste algoritmo é apresentada na secção 5.1.

Após a implementação do algoritmo, este foi utilizado num programa *MapReduce* com o objetivo de produzir os dois componentes do produto escolhido. Este programa é descrito na secção 5.2.

5.1 Implementação do algoritmo MLD

O algoritmo implementado para o cálculo da profundidade da camada mista é apresentado em [12]. A camada mista do oceano tem o seu limite na profundidade que tenha uma diferença de temperatura de 0.2 ºC, ou uma diferença de densidade potencial de 0.03 Kg/m^3, em comparação com a da superfície.

A implementação deste algoritmo teve como base o código-fonte fornecido pelos criadores do produto ArgoMixedLayers, que é constituída pelos seguintes passos:

- Cálculo da densidade potencial para cada medição de um perfil Argo.
- Encontrar a medição do perfil que ultrapassa o limite de temperatura / densidade potencial, comparando com a medição mais perto da superfície.
- Interpolar linearmente os valores de densidade potencial / temperatura com os de profundidade. São considerados como pontos de referência a medição à superfície e a medição que se encontra no limite encontrados.
- Encontrar o novo limite de temperatura e densidade potencial nos valores interpolados, e devolver a profundidade associada.

Para a implementação deste algoritmo foram ainda adaptadas para a linguagem Java, funções para o calculo da densidade potencial, provenientes da biblioteca PyOceans [1].

5.2 Implementação em MapReduce(SpatialHadoop/ST-Hadoop)

Após a implementação do algoritmo referido em 5.1, foi criado um programa *MapReduce* para produzir as duas componentes associadas ao caso de uso escolhido.

A implementação seguiu a estrutura apresentada na figura 5.1, e nas seguintes subsecções são apresentados os diferentes aspetos associados: **Input**, **Cálculo de MLD** e **Criação de climatologia**.

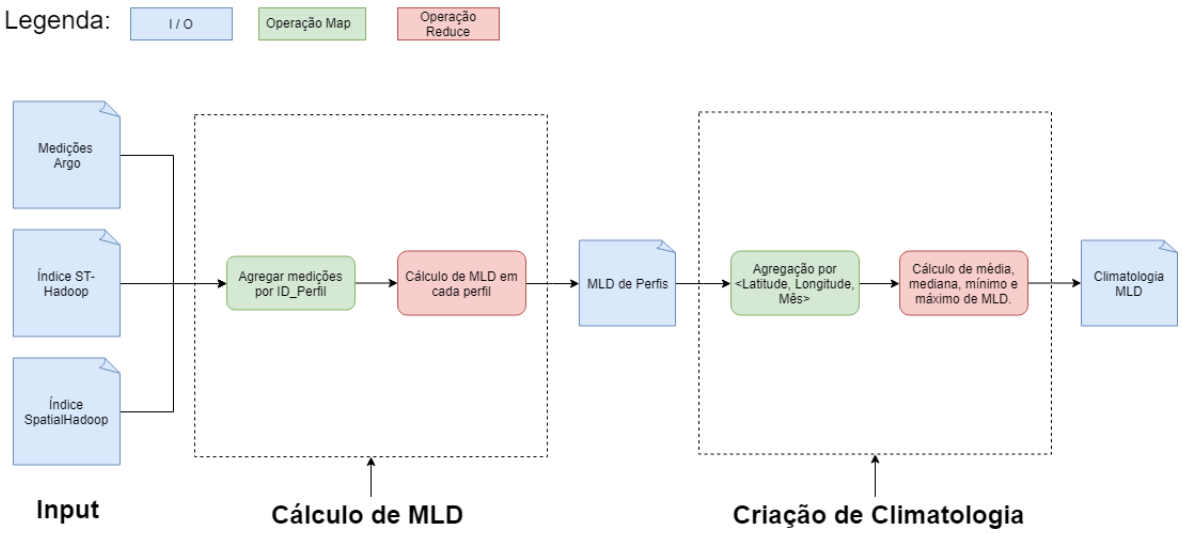


Figura 5.1: Esquema de implementação do produto ArgoMixedLayers em *MapReduce*

[1] https://pyoceans.github.io/python-oceans/, visitado a 02/2019

5.2.1 Input

Foram consideradas três formas de *input* para a operação: o ficheiro de medições Argo, um índice *SpatialHadoop* e um índice *ST-Hadoop*.

No caso em que se pretenda calcular a profundidade da camada mista para a totalidade dos perfis Argo, a utilização de índices espaciais providenciados pelos *frameworks* não trará melhoria de desempenho, visto que a totalidade do conjunto de dados terá de ser processado. Embora o uso dos *frameworks* neste caso não apresentem melhor desempenho, existem duas vantagens a considerar:

- Maior facilidade na criação de programas de processamento de dados espaciais devido aos tipos de dados e operações proporcionados pelo *framework*.
- Possibilidade de parametrização espacial de diferentes aplicações. Esta possibilidade, permite uma maior flexibilidade na criação do produto, já que é possível criá-lo para apenas uma área espacial / espaciotemporal considerada.

5.2.2 Cálculo de MLD

Para os diferentes dados de entrada considerados, o primeiro passo computacional consistiu no cálculo da profundidade da camada mista para cada perfil existente. Esta operação foi implementada como um programa *MapReduce* que recebe como parâmetro de entrada a área espacial /espaciotemporal a considerar.

Para dar uso dos índices espaciais / espaciotemporais, é necessário especificar o intervalo providenciado pelo utilizador e introduzi-lo nos parâmetros de execução do programa. Este intervalo será utilizado para escolher as partições do índice interceptadas:

```
1  CalcMLDMapReduce(Path inFile, Path outFile, OperationsParams params){
2      // Use the built-in range filter of the input format
3      params.set(SpatialInputFormat3.InputQueryRange, params.get("rect"));
4
5      //Create new MapReduce Job.
6      Job job = new Job(params, "Calc_MLD");
7      job.setJarByClass(Calc_MLD.class);
8
9      //Set input format (Recognize argo float data)
10     job.setInputFormatClass(SpatialInputFormat3.class);
11     SpatialInputFormat3.setInputPaths(job, inFile);
12
13     //Set map/reduce classes and output types
14     job.setMapperClass(CalcMLDMap.class);
15     job.setReducerClass(CalcMLDReduce.class);
16     job.setMapOutputKeyClass(Text.class);
17     job.setMapOutputValueClass(SProfile.class);
18
19     job.submit();
20 }
```

Listagem 5.1: Setup (Cálculo de MLD)

Como o conjunto de dados está armazenado por medições, foi primeiro necessário agregar todas as medições de um perfil. Este processo correspondeu à função **Map** do programa desenvolvido:

```
map(Rectangle cellMBR, Iterable<SProfile> value, Context context){
      for (SProfile s : value) {
        String id = s.getId();
        context.write(new Text(id), s);
      }
}
```

Listagem 5.2: Fase map (Cálculo de MLD)

Dado um intervalo espacial, cada **Map** recebe uma partição do índice e todas as medições que nele se encontram. É apenas necessário devolver um par (ID_PERFIL, medição) para que o **reducer** os receba organizados por ID_PERFIL, podendo assim utilizar as medições para calcular a profundidade da camada mista:

```
reduce(Text profile_id, Iterable<SProfile> value,Context context){
    double[] mlds;
    List<SProfile> measurements = new ArrayList<>();
    SProfile sp;

    //Order values by pressure/depth values (alternative: use secondary sorting)
    for (SProfile s : value) {
        measurements.add(s);
      }
    measurements.sort(Comparator.comparing(SProfile::getPressure));

    double[] temp = new double[measurements.size()];
    double[] sal = new double[measurements.size()];
    double[] press = new double[measurements.size()];

    //Organize measurement values in different arrays:
    for (int i = 0; i < measurements.size(); i++) {
        sp = measurements.get(i);
        temp[i] = sp.getTemperature();
        sal[i] = sp.getPSalinity();
        press[i] = sp.getPressure();
    }

    //Calculate profile MLD : returns [MDL_DENS, MLD_PRESS, TEMP, DENS]
    mlds = SWEquations.calc_mld(sal, temp, press);

    //Prepare output string
    sp = measurements.get(0);
    double latitude = sp.getLatitude();
```

```
30      double longitude = sp.getLongitude();
31      String time = sp.getTime();
32
33      String out = latitude + "," + longitude + "," + time +
34        "," + mlds[0] + "," + mlds[1] + "," + mlds[2] + "," + mlds[3];
35
36      output.set(out);
37      context.write(profile_id, output)
38  }
```

Listagem 5.3: Fase reduce (Cálculo de MLD)

O resultado desta operação, consiste num ficheiro em que cada linha representa um objeto espacial com as propriedades apresentadas na tabela 5.1.

Tabela 5.1: Variáveis resultantes do cálculo da profundidade da camada mista por perfil

Variável	Descrição
Id_perfil	Identificador do perfil Argo
Latitude	latitude do local de medição do perfil
Longitude	longitude do local de medição do perfil
Data	data de medição do perfil
MLD_Temp	Profundidade da camada mista, utilizando o limite de temperatura
MLD_Dens	Profundidade da camada mista, utilizando o limite de densidade potencial
Temperatura	Valor de temperatura na profundidade MLD_Temp
Densidade Potencial	Valor de densidade potencial na profundidade MLD_Dens

5.2.3 Criação de climatologia

Após a finalização do passo anterior, foi necessário calcular: a média, máximo e mínimo da profundidade da camada mista em cada bloco de $1°latitude \times 1°longitude \times$mês, para gerar a climatologia pretendida.

Foi criado outro processo *MapReduce*, em que a fase *Map*, mapeou cada linha do ficheiro resultante da operação anterior, num tuplo «Latitude,Longitude,Mes>, Linha>. O primeiro elemento corresponde a uma chave composta por Latitude, longitude e mês, e o valor é a própria linha a ser processada:

```
1  map(LongWritable line_number, MLD_Prof value, Context context){
2
3      //Get all key variables.
4      String month = value.getDate().split("/")[1]
5      double latitude = Math.round(value.getLatitude());
6      double longitude = Math.round(value.getLongitude());
7      CompositeKey outputKey = new CompositeKey(latitude,longitude, month);
8
9      context.write(outputKey, value);
10 }
```

Listagem 5.4: Fase map (Criação de climatologia)

A fase *Reduce* foi responsável por receber a informação agregada da profundidade da camada mista a cada ponto de latitude, longitude e mês do ano, e calcular os diferentes valores pretendidos sobre os dados da camada mista recebidos:

```
reduce(CompositeKey key, Iterable<MLD_Prof> value, Context context){
  double mld_temp_max, mld_temp_avg = 0.0;
  double mld_pden_max, mld_pden_avg = 0.0;
  double mld_pden_min, mld_temp_min = 0.0;
  double avg_temp, avg_pden = 0.0;
  int count = 0;

  for (MLD_Prof p : value) {
    double mld_pden = p.getMLDDens();
    double mld_temp = p.getMLDTemp();
    double temp = p.getTemp();
    double pden = p.getPden();

    if (mld_temp_max < mld_temp)
      mld_temp_max = mld_temp;
    if(mld_temp_min > mld_temp)
      mld_temp_min = mld_temp;

    if (mld_pden_max <mld_pden)
      mld_pden_max = mld_pden;
    if(mld_pden_min > mld_pden)
      mld_pden_min = mld_pden;

     avg_pden += pden;
     avg_temp += temp;
     mld_temp_avg += mld_temp;
     mld_pden_avg += mld_pden;
     count++;
  }

    String out = (mld_temp_avg / count) + "," + mld_temp_min + ","
    + mld_temp_max + (mld_pden_avg / count) + "," + mld_pden_min + ","
    + mld_pden_max + "," + (avg_temp / count) + "," (avg_pden / count)

  output.set(out);
  context.write(key, output)
}
```

Listagem 5.5: Fase reduce (Criação de climatologia)

Cada linha do ficheiro resultante contém as variáveis apresentadas na tabela 5.2.

Tabela 5.2: Variáveis resultantes do cálculo da climatologia

Variável	Descrição
Latitude	latitude do local de agregação de perfis
Longitude	longitude do local de agregação de perfis
Data	mês do ano
MLD_Temp_media	média da profundidade da camada mista, no limite de temperatura
MLD_Temp_min	mínimo da profundidade da camada mista, no limite de temperatura
MLD_Temp_max	máximo da profundidade da camada mista, no limite de temperatura
MLD_Dens_media	média da profundidade da camada mista, no limite de densidade potencial
MLD_Dens_media	mínimo da profundidade da camada mista, no limite de densidade potencial
MLD_Dens_media	máximo da profundidade da camada mista, no limite de densidade potencial
MLD_Dens	Média da densidade potencial da camada mista
MLD_Temp	Média da temperatura da camada mista

5.3 Conclusões

Neste capítulo foi pretendido demonstrar a implementação de um produto grelha, através do processamento de perfis Argo.

A solução foi desenvolvida através dos *frameworks* SpatialHadoop e *ST-Hadoop*, onde foi apresentado como é possível adaptar os tipos de dados e *queries* disponíveis, na implementação de um caso de uso específico. Os tipos de dados e operações presentes nestes *frameworks* facilitam o desenvolvimento de aplicações espaciais em contexto de *Big Data*.

O objetivo principal desta solução passou por mostrar a viabilidade dos *frameworks* analisados na resolução de problemas reais. Não houve uma avaliação pormenorizada do seu desempenho mas foi constatado que o tempo de execução na criação global do produto é comparável ao da criação de um índice espacial SpatialHadoop. A parte mais computacionalmente exigente é a referida na secção 5.2.2, visto ser necessário agregar todas as medições existentes no conjunto de dados.

O uso dos *frameworks* também apresentou alguma flexibilidade na criação do produto, sendo possível parametrizar a área espacial e temporal de criação. Isto poderá ser benéfico em contextos que a computação total de um produto apresente ser demasiado custosa.

Capítulo 6

Conclusões e Trabalho Futuro

Nesta dissertação foram concretizados quatro objetivos principais:

- Análise sistemática de diferentes caso de uso relacionados com perfis Argo.
- Caracterização das computações relacionadas com os casos de uso identificados.
- Experimentação de tecnologias Big Data para o processamento de dados espaciais/espaciotemporais.
- Implementação de um caso de uso real, utilizando as tecnologias avaliadas.

Para o trabalho de pesquisa feito sobre o projeto Argo, foi utilizado um método sistemático que consistiu na agregação dos diferentes artigos relacionados numa única base de dados. Esta base de dados foi fundamental para a melhor análise das aplicações mais relevantes associadas a perfis Argo, num curto espaço de tempo. Esta abordagem sistemática permitiu concluir que o uso de perfis Argo se divide em duas áreas: **Análise de fenómenos** e **Criação de produtos grelha**.

No trabalho experimental, existiu primeiro uma grande preocupação na escolha da infraestrutura a utilizar e na sua configuração ótima. Foi constatado que, em processamentos baseados em *Spark* ou *MapReduce*, para a melhor utilização dos recursos disponíveis num *cluster* existem diferentes parâmetros a ter em conta. No caso do *MapReduce* deve-se ter em conta o número de mappers e reducers que deverão ser executados por cada nó. No caso do *Spark* devem ser considerados quantos executores (incluindo a sua memória e cores CPU dedicadas) que deverão efetuar o processamento de dados e operações numa aplicação.

Na análise do estado da arte de tecnologias *Big data* para processamentos espaciais /espaciotemporais, foi verificada a importância da divisão e armazenamento eficiente dos

dados e em mecanismos de pesquisa que de melhor forma respondem a interrogações de foro espacial / espaciotemporal.

Na experimentação dos *frameworks Big Data* escolhidos, foi adotada uma abordagem sistemática de avaliação, onde foram considerados dois tipos diferentes de operações: Criação de índices e execução de *queries*. Foi possível avaliar o impacto que certos parâmetros têm na criação de índices espaciais / espaciotemporais em cada *framework*, e consequentemente, no desempenho de diferentes pesquisas espaciais/espaciotemporais nestes. Para além da avaliação geral a todas a *frameworks*, foram estudados outros aspetos particulares:

- No caso do SpatialHadoop, em que situações se deve utilizar uma implementação centralizada de uma determinada pesquisa ao invés MapReduce.
- Verificação da melhoria de desempenho de pesquisas espaciotemporais no ST-Hadoop em relação ao SpatialHadoop, e como este está relacionado com a capacidade do *cluster* responder a *queries* simultaneamente.
- Como a escolha correta do número de partições e do modo de persistência dos SRDD do Geospark, proporcionam a maior rapidez de processamento possível.

Por fim, foi implementado um caso de estudo específico em dois dos *frameworks* avaliados, algo que serviu para demonstrar o racional por detrás do desenvolvimento de programas sobre casos de uso reais utilizando estas tecnologias. Teria sido interessante explorar um caso de estudo mais complexo, mas grande parte do tempo despendido nesta dissertação focou-se: na compreensão das diferentes *frameworks*, na configuração ótima dos *clusters*, e no processo de avaliação experimental.

6.1 Trabalho futuro

Para trabalho futuro, será interessante explorar os seguintes pontos:

- Explorar o uso das tecnologias *Big Data* avaliadas neste documento em outras áreas de interesse. O uso destas tecnologias abrange todo o tipo de dados espaciais e poderão ser relevantes na implementação de soluções Big Data no processamento de outro tipo de dados espaciais / espaciotemporais.
- Explorar casos de uso de maior complexidade que relacionem perfis Argo com dados de outras fontes. Na análise de perfis Argo foi constatado que estes eram utilizados em conjuntos com dados de satélite ou outros dados marítimos. Seria interessante o uso das tecnologias estudadas nesta dissertação para responder a aplicações com maior exigência computacional.

Bibliografia

[1] *Argo Float Bibliography*. URL: http://www.argo.ucsd.edu/Bibliography.html.

[2] S. C. Riser, H. J. Freeland, D. Roemmich, S. Wijffels, A. Troisi, M. Belbéoch, D. Gilbert, J. Xu, S. Pouliquen, A. Thresher, P.-Y. L. Traon, G. Maze, B. Klein, M. Ravichandran, F. Grant, P.-M. Poulain, T. Suga, B. Lim, A. Sterl, P. Sutton, K.-A. Mork, P. J. Vélez-Belchí, I. Ansorge, B. King, J. Turton, M. Baringer e S. R. Jayne. "Fifteen years of ocean observations with the global Argo array". Em: *Nature Climate Change* 6.2 (fev. de 2016), pp. 145–153. DOI: 10.1038/nclimate2872. URL: https://doi.org/10.1038/nclimate2872.

[3] A. Wong, R. Keeley, T. Carval e A. D. M. Team. "Argo Quality Control Manual for CTD and Trajectory Data". Em: (2015), pp. –. DOI: 10.13155/33951.

[4] A. D. M. Team. "Argo user's manual V3.2". Em: (2017), pp. –. DOI: 10.13155/29825.

[5] D. Roemmich e J. Gilson. "The 2004–2008 mean and annual cycle of temperature, salinity, and steric height in the global ocean from the Argo Program". Em: *Progress in Oceanography* 82.2 (ago. de 2009), pp. 81–100. DOI: 10.1016/j.pocean.2009.03.004. URL: https://doi.org/10.1016/j.pocean.2009.03.004.

[6] S. Yi, W. Sun, K. Heki e A. Qian. "An increase in the rate of global mean sea level rise since 2010". Em: *Geophysical Research Letters* 42.10 (2015). 2015GL063902, pp. 3998–4006. ISSN: 1944-8007. DOI: 10.1002/2015GL063902. URL: http://dx.doi.org/10.1002/2015GL063902.

[7] G. McCarthy, D. Smeed, W. Johns, E. Frajka-Williams, B. Moat, D. Rayner, M. Baringer, C. Meinen, J. Collins e H. Bryden. "Measuring the Atlantic Meridional Overturning Circulation at 26°N". Em: *Progress in Oceanography* 130 (2015), pp. 91 –111. ISSN: 0079-6611. DOI: https://doi.org/10.1016/j.pocean.2014.10.006. URL: http://www.sciencedirect.com/science/article/pii/S0079661114001694.

[8] H. Fu, X. Wang, P. C. Chu, X. Zhang, G. Han e W. Li. "Tropical cyclone footprint in the ocean mixed layer observed by Argo in the Northwest Pacific". Em: *Journal of Geophysical Research: Oceans* 119.11 (2014), pp. 8078–8092. ISSN: 2169-9291. DOI: 10.1002/2014JC010316. URL: http://dx.doi.org/10.1002/2014JC010316.

[9] W.-Z. Zhang, H. Xue, F. Chai e Q. Ni. "Dynamical processes within an anticyclonic eddy revealed from Argo floats". Em: *Geophysical Research Letters* 42.7 (2015). 2015GL063120, pp. 2342–2350. ISSN: 1944-8007. DOI: 10.1002/2015GL063120. URL: http://dx.doi.org/10.1002/2015GL063120.

[10] *YoMaHa'07: Velocity data assessed from trajectories of Argo floats at parking level and at the sea surface*. URL: http://apdrc.soest.hawaii.edu/projects/yomaha/.

[11] J. Holte, L. D. Talley, J. Gilson e D. Roemmich. "An Argo mixed layer climatology and database". Em: *Geophysical Research Letters* 44.11 (jun. de 2017), pp. 5618–5626. DOI: 10.1002/2017gl073426. URL: https://doi.org/10.1002/2017gl073426.

[12] C. de Boyer Montégut, G. Madec, A. S. Fischer, A. Lazar e D. Iudicone. "Mixed layer depth over the global ocean: An examination of profile data and a profile-based climatology". Em: *Journal of Geophysical Research: Oceans* 109.C12 (2004). C12003, n/a–n/a. ISSN: 2156-2202. DOI: 10.1029/2004JC002378. URL: http://dx.doi.org/10.1029/2004JC002378.

[13] J. Holte e L. Talley. "A New Algorithm for Finding Mixed Layer Depths with Applications to Argo Data and Subantarctic Mode Water Formation". Em: *Journal of Atmospheric and Oceanic Technology* 26.9 (2009), pp. 1920–1939. DOI: 10.1175/2009JTECHO543.1. eprint: https://doi.org/10.1175/2009JTECHO543.1. URL: https://doi.org/10.1175/2009JTECHO543.1.

[14] J. Dean e S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". Em: *Commun. ACM* 51.1 (jan. de 2008), pp. 107–113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492. URL: http://doi.acm.org/10.1145/1327452.1327492.

[15] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker e I. Stoica. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing". Em: *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp. 15–28. ISBN: 978-931971-92-8. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

[16] R. A. Finkel e J. L. Bentley. "Quad trees a data structure for retrieval on composite keys". Em: *Acta Informatica* 4.1 (1974), pp. 1–9. DOI: 10.1007/bf00288933. URL: https://doi.org/10.1007/bf00288933.

[17] A. Guttman. "R-trees". Em: *ACM SIGMOD Record* 14.2 (1984), p. 47. DOI: 10.1145/971697.602266. URL: https://doi.org/10.1145/971697.602266.

[18] J. L. Bentley. "Multidimensional binary search trees used for associative searching". Em: *Communications of the ACM* 18.9 (1975), pp. 509–517. DOI: 10.1145/361002.361007. URL: https://doi.org/10.1145/361002.361007.

[19] A. Eldawy e M. F. Mokbel. "The Era of Big Spatial Data: A Survey". Em: *Foundations and Trends® in Databases* 6.3-4 (2016), pp. 163–273. DOI: 10.1561/1900000054. URL: https://doi.org/10.1561/1900000054.

[20] H. Tan, W. Luo e L. M. Ni. "CloST: A Hadoop-based Storage System for Big Spatio-temporal Data Analytics". Em: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: ACM, 2012, pp. 2139–2143. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2398589. URL: http://doi.acm.org/10.1145/2396761.2398589.

[21] Z. Li, F. Hu, J. L. Schnase, D. Q. Duffy, T. Lee, M. K. Bowen e C. Yang. "A spatio-temporal indexing approach for efficient processing of big array-based climate data with MapReduce". Em: *International Journal of Geographical Information Science* 31.1 (2017), pp. 17–35. DOI: 10.1080/13658816.2015.1131830. eprint: https://doi.org/10.1080/13658816.2015.1131830. URL: https://doi.org/10.1080/13658816.2015.1131830.

[22] A. Eldawy e M. F. Mokbel. "SpatialHadoop: A MapReduce framework for spatial data". Em: *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015. DOI: 10.1109/icde.2015.7113382. URL: https://doi.org/10.1109/icde.2015.7113382.

[23] A. Eldawy e M. Mokbel. "Pigeon: A spatial MapReduce language". Em: *Proceedings - International Conference on Data Engineering*. IEEE Computer Society, 2014, pp. 1242–1245. ISBN: 9781479925544. DOI: 10.1109/ICDE.2014.6816751.

[24] L. Alarabi, M. F. Mokbel e M. Musleh. "ST-Hadoop: A MapReduce Framework for Spatio-Temporal Data". Em: *Advances in Spatial and Temporal Databases*. Ed. por M. Gertz, M. Renz, X. Zhou, E. Hoel, W.-S. Ku, A. Voisard, C. Zhang, H. Chen, L. Tang, Y. Huang, C.-T. Lu e S. Ravada. Cham: Springer International Publishing, 2017, pp. 84–104. ISBN: 978-3-319-64367-0.

[25] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert e M. Ronquest. "GeoMesa: a distributed architecture for spatio-temporal fusion". Em: vol. 9473. 2015, pp. 9473 –9473 –13. DOI: 10.1117/12.2177233. URL: http://dx.doi.org/10.1117/12.2177233.

[26] J. Yu, Z. Zhang e M. Sarwat. "Spatial data management in apache spark: the GeoSpark perspective and beyond". Em: *GeoInformatica* 23.1 (2018), pp. 37–78. DOI: 10.1007/s10707-018-0330-9. URL: https://doi.org/10.1007/s10707-018-0330-9.

[27] A. Eldawy, L. Alarabi e M. F. Mokbel. "Spatial partitioning techniques in SpatialHadoop". Em: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 1602–1605. DOI: 10.14778/2824032.2824057. URL: https://doi.org/10.14778/2824032.2824057.

[28] G. Shurkhovetskyy, N. Andrienko, G. Andrienko e G. Fuchs. "Data Abstraction for Visualizing Large Time Series". Em: *Computer Graphics Forum* (), n/a–n/a. ISSN: 1467-8659. DOI: 10.1111/cgf.13237. URL: http://dx.doi.org/10.1111/cgf.13237.

[29] J. Lin, E. Keogh, L. Wei e S. Lonardi. "Experiencing SAX: a novel symbolic representation of time series". Em: *Data Mining and Knowledge Discovery* 15.2 (2007), pp. 107–144. ISSN: 1573-756X. DOI: 10.1007/s10618-007-0064-z. URL: https://doi.org/10.1007/s10618-007-0064-z.

[30] E. Keogh, K. Chakrabarti, M. Pazzani e S. Mehrotra. "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases". Em: *Knowledge and Information Systems* 3.3 (2001), pp. 263–286. ISSN: 0219-1377. DOI: 10.1007/PL00011669. URL: https://doi.org/10.1007/PL00011669.

Anexo I

Preparação de Trabalho Experimental

Neste capítulo é apresentada a abordagem e preparação do trabalho experimental realizado. A preparação deste trabalho divide-se nos seguintes pontos:

Escolha de frameworks Big Data (??): Na secção 3.3 foram apresentadas diferentes *frameworks* que permitem o processamento de dados espaciotemporais no contexto de tecnologias Big Data. De todas elas, foram escolhidas três para avaliar tendo em conta critérios técnicos e de usabilidade.

Configuração de Infraestrutura (I.1): Após tomadas as decisões no ponto anterior, foram contempladas várias opções de infraestrutura para suportar o processamento dos perfis Argo. Foram também estudadas as melhores opções de configuração da infraestrutura escolhida.

I.1 Configuração de Infraestrutura Big Data

A escolha e configuração de infraestrutura capaz de processar uma grande dimensão de dados foi crucial para o trabalho desenvolvido.

Numa primeira fase, foi utilizado um pequeno *cluster* constituído por 3 *Raspberry Pis*, montado e configurado especificamente para este trabalho. Cada *raspberry pi* contém um cpu com 4 cores, e 1GB de RAM. Pela facilidade nas mudanças de configuração, esta infraestrutura serviu o propósito de realizar o processo de instalação e testes preliminares nos *frameworks Big Data* escolhidas para avaliar experimentalmente, bem como para dar uma ideia concreta sobre o efeito de diferentes parâmetros de configuração no processamento de dados, sem grande dispêndio de tempo. O uso desta infraestrutura foi importante para o processo de configuração que acabou por ser utilizado no *cluster* onde foi realizado o

trabalho experimental (apresentado nas secções I.2.1 e I.2.2) . A imagem I.1 , apresenta o *cluster* de *Raspberry Pis* utilizado.

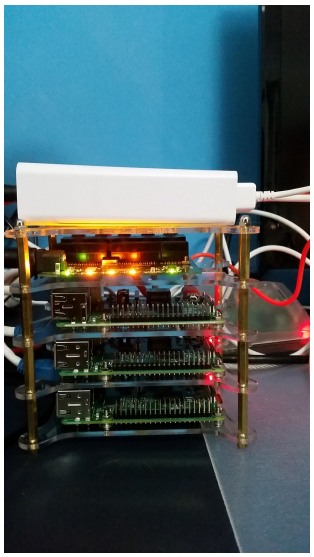


Figura I.1: Cluster Raspberry Pi utilizado em testes preliminares.

A necessidade de uma infraestrutura mais capaz levou à procura de opções mais robustas, e foram equacionadas duas soluções Cloud: HDInsight[1] e ElasticMapReduce[2]. Também era possível a utilização de um *cluster* pertencente ao Departamento de Informática da FCT UNL que obrigava a utilização de Docker e Docker Swarm para a execução do software.

As soluções *Cloud*, embora sejam as mais robustas e de fácil configuração, apresentaram algumas particularidades que dificultaram o trabalho experimental:

Custo associado Ambas as plataformas têm um custo de utilização associado, fator que condiciona a flexibilidade no processo experimental em relação ao uso de uma infraestrutura local.

Armazenamento de dados Na solução *HDInsight*, o método de armazenamento utilizado, que é uma extensão do HDFS (Windows Azure Blob Storage), e não foi encontrada forma de alterar o tamanho de bloco do HDFS, propriedade importante a ser utilizada na avaliação experimental dos *frameworks*.

Limitação de máquinas No caso da solução ElasticMapReduce, o maior entrave encontrou-se nas limitações ao número de máquinas a utilizar (seria necessário pedir uma expansão destes limites) em comparação com a infraestrutura local disponível.

A infraestrutura que acabou por ser escolhida para o trabalho experimental descrito nos capítulos 5 e 6 deste documento, consistiu em 13 nós de computação pertencentes ao

[1] https://azure.microsoft.com/pt-pt/services/hdinsight/, visitado a 02/2019
[2] https://aws.amazon.com/emr/, visitado a 02/2019

Departamento de Informática da FCT-UNL. Nas seguintes subsecções estão descritas as diferentes opções de configuração necessárias para o seu melhor aproveitamento.

I.2 Configuração física

A infraestrutura utilizada no trabalho experimental está dividida em três grupos de configuração diferentes. Para processamentos utilizando os *frameworks* baseadas em *MapReduce* foram utilizados os nós pertencentes ao grupo 1 e 2. Para processamentos utilizando *frameworks* baseadas em *Spark*, foram utilizados nós do grupo 3. Esta divisão ocorreu porque os nós do grupo 3 permitem obter um melhor aproveitamento em computações em *Spark* devido à elevada RAM associada a cada nó.

A configuração dos diferentes grupos está apresentada na tabela I.1.

Tabela I.1: Configuração física dos nós disponíveis na infraestrutura utilizada.

	Grupo1	Grupo2	Grupo3
RAM	8GB	8GB	64GB
CPU (Cores)	8	16	24
Nº de Máquinas	6	3	3

I.2.1 Cluster Hadoop(MapReduce)

Para computações MapReduce, a configuração do *cluster* baseou-se num *script* disponibilizado pela Hortonworks [3] que determina os diferentes parâmetros de alocação de memória ótimos, através da configuração física de cada máquina disponível. Os parâmetros relevantes e o método de cálculo dos seus valores estão apresentados na tabela I.2.

O valor da variável *RAM-PER-CONTAINER* depende apenas da memória disponível em cada nó. A variável *containers* indica o número máximo de *containers* YARN que estão disponíveis em cada nó. O controlo sobre o número de *containers* disponíveis por nó é feito pelas primeiras 4 variáveis presentes na tabela. Os restantes parâmetros servem para indicar qual a memória que deve ser alocada por container em operações *MapReduce*, estando em conformidade com os limites estabelecidos nas primeiras variáveis. Por exemplo, a configuração ótima para nós do grupo 1 (referenciado na tabela I.1) consiste em ter no máximo três *containers* a executar em simultâneo por nó. Por exemplo, em programas *MapReduce*, isto significa que poderão estar no máximo 3 execuções de uma operação *Map* em simultâneo por nó.

Para além da preparação referida, foi também necessário preparar a infraestrutura utilizada para a execução de programas *MapReduce* em simultâneo. Existem dois parâmetros a ter em conta para atingir esse efeito que estão apresentados na tabela I.3.

Como foi referido na secção 3.1.2, cada aplicação que seja executada num cluster gerido pelo YARN tem um *Application Master* associado. Os parâmetros apresentados

[3] https://hortonworks.com, visitado a 02/2019

Tabela I.2: Parâmetros de configuração do cluster *MapReduce*.

Parâmetro	Valor
yarn.nodemanager.resource.memory-mb	= containers * RAM-per-container
yarn.scheduler.minimum-allocation-mb	= RAM-per-container
yarn.scheduler.maximum-allocation-mb	= containers * RAM-per-container
mapreduce.map.memory.mb	= RAM-per-container
mapreduce.reduce.memory.mb	= 2 * RAM-per-container
mapreduce.reduce.java.opts	= 0.8 * 2 * RAM-per-container

Tabela I.3: Configurações de execução de programas simultâneamente

Parâmetro	Valor
yarn.scheduler.capacity. maximum-am-resource-percent	Percentagem máxima de memória dedicada a Application Masters
yarn.app.mapreduce.am.resource.mb	Memória a alocar a cada Application Master

controlam a percentagem máxima de memória possível a alocar a todos os *Application Masters*, e a memória que cada *AM* poderá utilizar respetivamente. A definição destes parâmetros controla efetivamente o número máximo de *Application Masters* que poderão estar ativos ao mesmo tempo.

É também necessário ter em conta que uma definição deficiente destes parâmetros irá impactar o desempenho da infraestrutura. Um número demasiado elevado de *AM* poderá causar o efeito de *starvation* (falta de memória), e um número demasiado baixo poderá não aproveitar todos os recursos disponíveis.

A definição dos parâmetros foi feita manualmente, ao realizar um número concorrente das operações pretendidas a avaliar com diferentes valores considerados. Foi concluído que utilizar no máximo 80% da memória do cluster e reservar 1.35GB de memória para cada *AM*, apresentaram ser as melhores configurações para as computações associadas.

I.2.2 Cluster Spark

Para computações distribuídas em *Spark*, um aspeto importante é a especificação do número de executores e cores (CPU) que estão designados a uma aplicação. Estes dois elementos podem ser especificados através de opções de configuração providenciadas em tempo de execução. As opções relevantes estão apresentadas na tabela I.4.

Tabela I.4: Configurações de execução de programas Spark.

Parâmetro	Valor
num-executors	Define o número de executores da aplicação.
executor-cores	Define o número de cores(CPU) dedicadas a cada executor
executor-memory	Define a memória dedicada a cada executor.

Para otimizar a performance do *cluster*, foram seguidas algumas diretrizes[4] que indicam que um executor deverá ter no máximo 64GB de memória e 5 cores alocadas. Tendo isto e as características físicas do cluster a ser utilizado (nós pertencentes ao grupo 3) em conta, foi possível determinar os valores ótimos para as diferentes variáveis de configuração.

Para além dos parâmetros de configuração acima descritos, foi necessário alterar os parâmetros de memória YARN (primeiros três parâmetros na tabela I.2) para estarem de acordo com a memória atribuída a cada executor. Por fim, foi ainda necessário utilizar o *DominantResourceCalculator* presente no YARN porque o gestor de recursos por-defeito apenas tem em conta a memória na atribuição de *containers* a aplicações, não aproveitando todas as cores CPU disponíveis.

[4] http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-2/, visitado a 02/19

Anexo II

Séries Temporais

Uma série temporal consiste numa coleção de observações, obtidas através de medições sequenciais ao longo do tempo. Vários domínios como o financeiro, médico e cientifico produzem uma grande quantidade de dados expressos sob a forma de séries temporais. Devido ao aumento sistemático da dimensão deste tipo de dados, surgiu a área de *Time SeriesData Mining*, área de estudo que se foca na extração de informação eficiente de grandes séries temporais.[28]

Grande parte dos estudos que utilizam perfis Argo, integram uma vertente temporal. Através dos perfis medidos em determinadas áreas geográficas é possível obter a evolução de diferentes variáveis como a salinidade e temperatura ao longo do tempo. Estas séries temporais poderão atingir uma grande dimensão, o que motivou o estudo sobre a área de TSDM.

II.1 Representações de Séries Temporais

A análise de séries temporais de grande dimensão (que contém um elevado número de observações) apresenta uma certa dificuldade devido à ineficiência de algoritmos específicos para a extração de diferentes tipos de informação. Para colmatar esse problema, existe a necessidade de encontrar uma abstração sobre a série temporal de forma a reduzir a sua dimensão, mas mantendo todas ou algumas das suas características.

Em [29] é apresentada a seguinte taxonomia de representações temporais:

A taxonomia divide em primeiro lugar as representações por serem *Data Adaptive* ou *Non Data Adaptive*. Representações *Data adaptive* contém parâmetros de entrada que permitem um melhor ajuste dependendo dos dados em questão, enquanto que representações *Non Data adaptive* a transformação da série temporal original é executada sempre da mesma forma independentemente dos dados em questão. No restante da hierarquia,

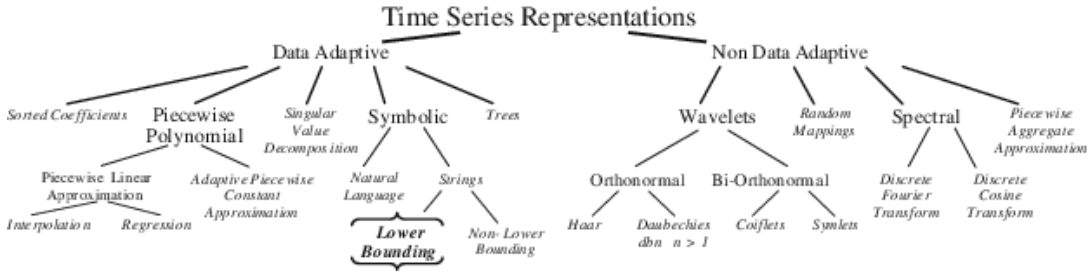


Figura II.1: Taxonomia de representações de séries temporais.[29]

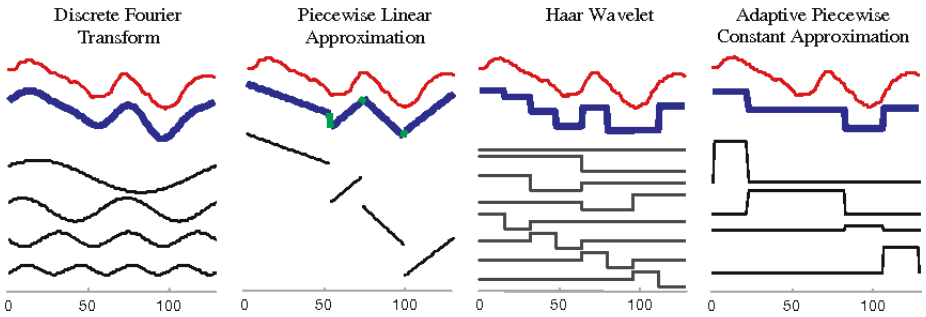


Figura II.2: Diferentes abordagens para a representação de séries temporais.[29]

estão descritos os métodos de representação, que no geral podem ser vistos como a tentativa de aproximação da série temporal original, através de um conjunto linear de uma função base. A figura II.2 apresenta quatro abordagens diferentes de representações de séries temporais, cada uma baseada num conjunto de funções base diferente.

A taxonomia apresentada na figura II.1 mostra apenas uma visão algorítmica das diferentes representações. Mais recentemente em [28] é procurada uma classificação das diferentes representações existentes tendo em conta as propriedades inerentes das séries temporais a serem transformadas como também as propriedades que são mantidas na nova representação:

Dimensionalidade A Série temporal é uni-variável ou multi-variável. Séries temporais uni-variáveis têm em conta apenas uma variável ao longo do tempo, enquanto que multi-variáveis contém 2 ou mais.

Disponibilidade A série está completamente disponível à partida ou através de *streaming*.

Preservação de características Identificar quais as características que são preservadas na nova representação.

Todos os aspetos apresentados devem ser bem analisados na escolha de uma representação, algo que é importante na análise de séries temporais. Para a sua escolha deverão estar bem definidos os objetivos pretendidos e as propriedades dos dados a serem analisados, de forma a escolher a representação mais adequada.

No restante desta secção, são apresentadas algumas representações estudadas e as suas propriedades:

II.1.1 Piecewise Aggregate Approximation

Piecewise Aggregate Approximation[30] aproxima uma série temporal com N elementos num vetor com M elementos. A série temporal é dividida em M partes iguais e é calculado o valor médio de cada parte. A representação reduzida da série será o conjunto de médias calculadas para cada parte.

A figura II.3 apresenta a redução de uma série temporal(gerada de forma aleatória) com 48 elementos com M = 8. A série original é divida em segmentos contendo 6 valores, são agregados e é calculada a sua média(linha vermelha). A nova representação da série temporal será o conjunto de médias calculadas, sendo a nova representação da série ter apenas 8 elementos.

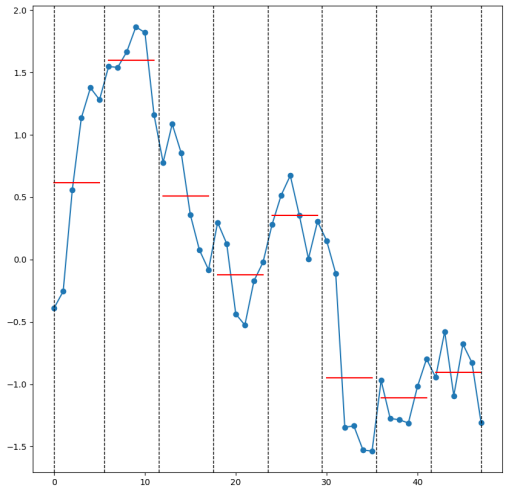


Figura II.3: Aplicação de PAA, com nº de segmentos = 8, a uma série temporal com 48 elementos.

II.1.2 Symbolic Aggregate Aproximation(SAX)

Symbolic Aggregate Approximation[29] consiste na transformação de uma série temporal num conjunto símbolos. Este algoritmo requer a escolha de dois parâmetros: O tamanho do alfabeto a ser utilizado na transformação(W) e M que corresponde ao número de divisões a serem consideradas na aplicação de PAA. Esta transformação é caracterizada por 3 passos distintos:

- a série temporal é z-normalizada de forma a ter uma média de aproximadamente 0 e desvio padrão de aproximadamente 1. O processo de normalização é feito através

B \W	3	4	5
B1	-0,43	-0.67	-0.84
B2	0.43	0	-0.25
B3		0.67	0.25
B4			0.84

Tabela II.1: Pontos de quebra para alfabetos contendo entre 3 a 5 elementos.

da seguinte fórmula:

$$x'_i = \frac{x_i - \mu}{\sigma}$$

A cada ponto da série temporal é subtraída a média, e o resultado da diferença é divido pelo desvio padrão.

- De seguida, a dimensão da série temporal é reduzida através da aplicação de PAA sobre a série temporal normalizada(explicado em II.1.1).

- Após ser obtida a série temporal reduzida, esta é transformada em símbolos de forma equiprovável. Este método assume que séries temporais normalizadas assumem uma distribuição normal. É então feita a distribuição de símbolos através da definição de pontos de quebra numa curva gaussiana *N(0,1)*, de forma a ser dividida em áreas de igual parte, dependendo do valor de W fornecido. a tabela II.1 apresenta os diferentes pontos de divisão para alfabetos entre 3 a 5 elementos, e a figura II.4 apresenta a transformação de uma série temporal com 48 elementos na palavra "dedcdaaa"utilizando os pontos de quebra apresentados(linhas a tracejado).

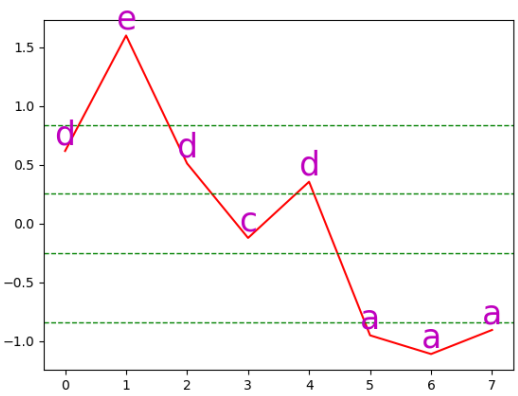


Figura II.4: Aplicação de SAX, com tamanho de N = 8 e W = 4, a uma série temporal com 48 elementos.

II.2 Operações sobre séries Temporais

Um aspeto a ser estudado quando se discutem séries temporais está relacionado com o tipo de pesquisas que se podem efetuar sobre elas. Estudos como [28] e também mais recentemente [28] definem o seguinte conjunto de operações:

Pesquisa Dada uma série temporal Q e uma medida de similaridade D(Q,C), ser possível encontrar a série temporal mais semelhante.

Classificação Atribuição de uma série temporal a uma classe, dado um conjunto pré-definido de classes.

Agrupamento Dado um conjunto de séries temporais, encontrar o melhor conjunto de grupos.

Deteção de Anomalias Dada uma série temporal e um modelo de comportamento normal, encontrar secções da série temporal que não correspondem ao comportamento esperado.

Deteção de Padrões Encontrar sequências comuns em séries temporais.

Visualização Possibilidade de visualizar séries temporais de grande dimensão.

As operações apresentadas estão descritas de forma genérica, o seu significado e importância será dependente do contexto da série temporal.